Tobit Flatscher

Lattice-Boltzmann Method for Multi-component Flows in Porous Media

Master's Thesis

to achieve the university degree of

Diplomingenieur (Master of Science)

Master's degree programme: Mechanical Engineering

submitted to

Graz University of Technology

Supervisor

Dipl-Ing. Dr.techn. René Prieler

Institute of Thermal Engineering
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Christoph Hochenauer

Graz, December 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Abstract

The main topic of this thesis is the *Lattice-Boltzmann methods* (LBM), an emerging variety of methods that can be applied to *computational fluid dynamics* (CFD) and can be seen as stylised versions of the Boltzmann equation. The resulting simple molecular dynamics algorithms underlie a complex theoretical and mathematical framework offering at the same time unforeseeable possibilities. Therefore this field is highly researched at the moment with thousands of papers being published every year.
In this thesis first a *single component* 2D and 3D model for *incompressible fluid flow* with *D*2*Q*9 and *D*3*Q*19 discretisation and BGK- as well as TRT-collision operator is created using C++ and later extended to *multi-component flow* using an advection-diffusion model based on Fick's law. Finally in order to boost the stability of the method a *Smagorinsky turbulence model* is included. All models are verified using standard benchmark scenarios for single component and binary flow and are eventually applied to laminar flow in *realistic porous media* obtained by tomography scans in order to demonstrate the main advantages of these models over traditional methods. Particular effort is put into parallelising the code on a multi-core processor using *OpenMP* in order to boost the computational speed.
The results for the advection-diffusion simulation in a porous bed made of random spheres show excellent agreeance with an Ansys Fluent simulation while the simulation of the realistic porous medium yields plausible results and will be further investigated in future work. The resulting model is shown to have significant advantages regarding speed and modelled phenomena over traditional methods in particular for transient flows in complex geometries and is even able to compete in terms of computational speed and parallel scaling with LBM-based open-source implementations such as Palabos. On a twelve-core processor the implemented algorithm is able to update 295 million cells per second for the *D*2*Q*9 lattice and 100 million cells for the *D*3*Q*19 lattice.

Acknowledgement:
This work was supported by the *Werner Hochegger Stiftung*.

Contents

Abstract **iii**

List of acronyms and nomenclature **xiii**

1. Acronyms . . . xiii
2. Nomenclature . . . xiv
   - 2.1. Greek symbols and special characters . . . xiv
   - 2.2. Roman symbols . . . xvi
3. Figures and diagrams . . . xxi
   - 3.1. Curves and diagrams . . . xxi
   - 3.2. Discrete distribution functions . . . xxi

Preface **xxv**

1. A brief introduction to computational fluid dynamics **2**

- 1.1. Fluid mechanics . . . 2
  - 1.1.1. Conservation equations for compressible flow . . . 2
  - 1.1.2. Speed of sound and Mach number . . . 5
  - 1.1.3. Conservation equations for incompressible flow . . . 7
  - 1.1.4. Turbulent flow and Kolmogorov microscales . . . 8
- 1.2. Computational fluid dynamics . . . 10
  - 1.2.1. Boundary conditions . . . 11
  - 1.2.2. Turbulence modelling . . . 12
  - 1.2.3. Insufficiencies of conventional methods . . . 13

2. The Lattice-Boltzmann Method **15**

- 2.1. The theory behind Lattice-Boltzmann . . . 15
  - 2.1.1. Kinetic theory of gases . . . 15
  - 2.1.2. Lattice gas automata . . . 16
  - 2.1.3. Statistical mechanics . . . 18
  - 2.1.4. Maxwell-Boltzmann distribution . . . 19
  - 2.1.5. Boltzmann equation . . . 19
- 2.2. Incompressible LBM . . . 22
  - 2.2.1. Derivation of incompressible Lattice-Boltzmann . . . 23
  - 2.2.2. Non-dimensionalisation . . . 27
  - 2.2.3. The algorithm . . . 28
  - 2.2.4. Discretisations . . . 30
  - 2.2.5. Macroscopic quantities . . . 32
  - 2.2.6. Errors and accuracy . . . 32
  - 2.2.7. Stability . . . 34
  - 2.2.8. Initial conditions . . . 35
  - 2.2.9. Boundary conditions . . . 35
  - 2.2.10. Forces on structures . . . 42
  - 2.2.11. Turbulence models . . . 43

2.2.12. Advantages over conventional CFD 44

3. Multi-component flows 45
3.1. Multi-component and multi-phase flows 45
3.2. Diffusion 46
3.2.1. Fick's laws 46
3.3. Advection and diffusion 47
3.3.1. Schmidt and Péclet number 47

4. LBM for multi-component flows 49
4.1. Advection-diffusion model 49
4.1.1. Accuracy, errors and correction terms 50
4.1.2. Stability and TRT collision operator 51
4.1.3. Conversion to physical units 51
4.1.4. Reduced lattices 51
4.1.5. Boundary conditions 52
4.1.6. Advantages of LBM for advection-diffusion 54

5. Porous media 55
5.1. Pore size, porosity and tortuosity 55
5.2. Particle Reynolds number 55
5.3. Pressure drop in packed beds 56
5.3.1. Carman-Kozeny equation 56
5.3.2. Ergun equation 56
5.4. Diffusion mechanisms in porous media 57
5.4.1. Effective diffusion coefficient 57
5.5. Residence time 57
5.5.1. Residence time distribution 58
5.5.2. Cumulative residence time distribution 59
5.5.3. Experimental measurement 59
5.6. Porous LBM: Sparse domain optimisation 60

6. Implementation 61
6.1. Single component flow 61
6.1.1. Test case 1: Lid-driven cavity 61
6.1.2. Stability of collision operators 63
6.1.3. Test case 2: Flow around a cylinder 64
6.1.4. Multi-component flow 70
6.2. Performance 71
6.2.1. Benchmark system 72
6.2.2. Matlab 72
6.2.3. C++ 73
6.3. Porous bed - comparison to Ansys Fluent 82
6.3.1. Geometry generation and simulation setup 82
6.3.2. LBM: mesh generation and settings 82
6.3.3. Fluent: mesh generation and simulation setup 83
6.3.4. Porosity and tortuosity 84
6.3.5. Pressure drop 84
6.3.6. Cumulative residence time distribution 85
6.3.7. Simulation runtime 86

7. Application: Real porous media **88**
7.1. Computational domain 88
7.2. Simulation setup 89
7.3. Results 89
7.3.1. Porosity, tortuosity and diffusion coefficient 89
7.3.2. Pressure drop 90
7.3.3. Residence time distribution 91

8. Conclusion and Outlook **93**
8.1. Conclusion 93
8.2. Physical outlook: reactive multi-component flow 94
8.3. Computational outlook: GPGPU and MPI 95

A. Thermodynamics **97**
A.1. State and process variables 97
A.2. Internal energy 97
A.3. The first law of thermodynamics for a closed system 98
A.4. Ideal gas 98
A.4.1. Perfect gas 98
A.5. Enthalpy and heat capacities 99
A.6. Entropy 100

B. Fluid dynamics **101**
B.1. Vorticity and vortex detection 101
B.1.1. Vortex detection 101
B.2. Conservation equations in integral notation 102
B.3. Conservative and non-conservative form 102
B.3.1. Non-conservative momentum equation 102
B.3.2. Non-conservative energy equation 103
B.4. Other important dimensionless numbers 103
B.5. Euler and Lagrange specification 103
B.6. Incompressible flow 103
B.6.1. Divergence free velocity field 103
B.6.2. Derivation of the Poisson's equation for pressure 104
B.7. Relations of density, pressure and temperature and Mach number . . . 104
B.7.1. Flow regimes 105
B.8. Inviscid flow: Euler equations 105
B.9. Bernoulli equation 106
B.10. Fluid structure interactions 106
B.10.1. Pressure coefficient 106
B.10.2. Drag coefficient 107
B.10.3. Lift coefficient 108
B.11. Strouhal number 108

C. Kinetic theory of gases **109**
C.1. Basic measures of kinetic theory 109
C.2. A simplified kinetic model 109
C.2.1. Pressure 110
C.2.2. Thermal energy 110
C.3. Particle distributions 111
C.3.1. Equilibrium 111

C.3.2. Derivation of the Maxwell-Boltzmann equilibrium distribution . 111
C.3.3. Moments of particle distributions 112
C.3.4. Collision integral: Stoßzahlansatz 113
C.3.5. Dimensionless Boltzmann equation 113

D. Multi-component flow 115
D.1. Basic definitions for mixtures 115
D.1.1. Scale of separation and homogeneous mixtures 115
D.2. Mole, mass and volume fractions 115
D.2.1. Mole fractions 116
D.2.2. Mass fractions 116
D.2.3. Molar concentration, partial volume and volume fractions 116
D.2.4. Mixture averages 116
D.3. Total, convective and diffusive flux 117
D.4. Fick's laws 118
D.4.1. Derivation of Fick's first law 118
D.4.2. Derivation of Fick's second law 118
D.4.3. Correction velocity 119
D.5. Lewis number 119
D.6. Non-binary multi-component flow 119
D.7. Maxwell-Stefan system 120
D.7.1. Derivation of the Maxwell-Stefan system for binary flow 120
D.7.2. Derivation of Fick's first law from the Maxwell-Stefan system . . 121

E. Lattice-Boltzmann 122
E.1. Derivation of Lattice-Boltzmann 122
E.1.1. Discrete Lattice-Boltzmann 122
E.1.2. Chapman-Enskog expansion 127
E.2. Force terms 131
E.3. Energy equation 132

F. LBM beyond incompressible flow 133
F.1. Derivation of Lattice-Boltzmann algorithms 133
F.1.1. Finite-difference discretisations 134
F.1.2. Finite-volume discretisations 134
F.2. Applications 134
F.3. Generic advantages 135

G. Computer architecture and programming 136
G.1. Basics of computer architecture 136
G.1.1. Virtual memory: Stack and heap 136
G.1.2. Processor architecture and word size 137
G.1.3. Processor clock rate and Intel turbo boost 137
G.1.4. Memory architecture 137
G.1.5. Multi-tasking and multi-threading 139
G.2. Interpreted and compiled implementations 139
G.3. Matlab 140
G.4. C++ 140
G.4.1. Compilation in C++ and compiler settings 140
G.4.2. Object-oriented programming 141
G.4.3. Passing by value and by reference 141

G.4.4. Choice of containers . . . 142
G.4.5. Inline functions . . . 142
G.4.6. Other optimisations . . . 142
G.4.7. Parallelisation and multi-threading . . . 143
G.4.8. Amdahl's law . . . 145
G.4.9. OpenMP . . . 145
G.4.10. A note on platform dependencies and portability . . . 145

H. Benchmarks **147**
H.1. Lid driven cavity: Pressure and vorticity . . . 147
H.2. Multi-component flow: Effects of wrong simulation parameters . . . 148

I. Mathematical appendix **150**
I.1. Mathematical notations . . . 150
I.1.1. Scalars . . . 150
I.1.2. Vectors . . . 150
I.1.3. Matrices . . . 151
I.1.4. Differentials . . . 151
I.1.5. Einstein notation . . . 151
I.1.6. Kronecker delta . . . 152
I.2. Probability . . . 152
I.2.1. Gaussian distribution function . . . 152
I.3. Integrals and derivatives . . . 152
I.3.1. Material derivative . . . 152
I.3.2. Gauss's divergence theorem . . . 152
I.3.3. Important integrals . . . 153
I.3.4. Orthogonal functions . . . 153
I.3.5. Gaussian quadrature . . . 153
I.4. Series expansions . . . 154
I.4.1. Taylor series . . . 155
I.4.2. Laurent series . . . 155
I.4.3. Fourier Series . . . 155

J. Perturbation theory **158**
J.1. Examples . . . 158
J.1.1. Example 1: Transcendental equations . . . 159
J.1.2. Example 2: Differential equations . . . 159
J.2. Two timing multiple scale method . . . 161
J.2.1. Derivatives of multiple-scale perturbation series . . . 162

Bibliography **163**

List of Figures

1. Pre- and post-collision populations . . . xxi
2. Solid and fluid nodes . . . xxii

1.1. Possible deformations of a continuum element . . . 4
1.2. Speed of sound . . . 5
1.3. Mach number . . . 6
1.4. Density ratio over Mach number . . . 8
1.5. Laminar and turbulent flow . . . 9
1.6. Equation systems in CFD . . . 10
1.7. Grids in CFD . . . 10
1.8. Wall function . . . 11
1.9. Reynolds averaging . . . 12
1.10. Classification of CFD methods . . . 14

2.1. Knudsen number . . . 16
2.2. LGA algorithm . . . 16
2.3. Non-deterministic collision rules in LGA . . . 17
2.4. Lattice gas automata: HPP and FHP . . . 17
2.5. Maxwell-Boltzmann distribution . . . 19
2.6. Scales of fluid flow . . . 23
2.7. Schematic Lattice-Boltzmann derivation . . . 26
2.8. Shock waves in LBM . . . 26
2.9. LBM lattices: D2Q9 and D3Q19 . . . 28
2.10. LBM algorithm . . . 29
2.11. Schematic LBM algorithm . . . 30
2.12. Possible orientations of D2Q9 . . . 31
2.13. Possible orientations of D3Q19 . . . 31
2.14. Error sources in LBM . . . 33
2.15. LBM initialisation . . . 35
2.16. Stair-cased boundaries . . . 35
2.17. Link-wise and wet-node boundary conditions . . . 36
2.18. Periodic boundaries . . . 37
2.19. No-slip wall: full-way bounce-back . . . 38
2.20. No-slip wall: half-way bounce-back . . . 38
2.21. Free-slip wall . . . 39
2.22. Corners in $2D$ . . . 42
2.23. Corners and edges in $3D$ . . . 42
2.24. Momentum exchange algorithm . . . 42

3.1. Multi-component and multi-phase flow . . . 45
3.2. Advection-diffusion . . . 47

4.1. Reduced lattices: $D2Q5$ and $D3Q7$ . . . 51

4.2. Anti-bounce back Dirichlet boundary 52

5.1. Residence time distribution 58
5.2. Experimental RTD 59

6.1. Lid-driven cavity 61
6.2. Deep LDC: streamlines 62
6.3. LDC 3D: velocity cross-section 62
6.4. Square LDC: streamlines 63
6.5. Square LDC: velocity cross sections 63
6.6. Stability of collision operators 64
6.7. Flow around a cylinder 64
6.8. Flow regimes around a cylinder 65
6.9. Strouhal number for circular cylinders 66
6.10. Blockage ratio and Strouhal number 66
6.11. Cylinder: pressure distribution 67
6.12. Cylinder: vorticity 67
6.13. Cylinder: pressure 68
6.14. Benchmark: Strouhal number 68
6.15. Cylinder: pressure coefficient 69
6.16. Cylinder: drag and lift coefficient 69
6.17. Vorticity around 3D cylinder 69
6.18. Gaussian hill 70
6.19. Gaussian hill diffusion 71
6.20. Gaussian hill advection 71
6.21. Memory layout 73
6.22. Simple parallel loops 73
6.23. Schematic memory allocation 74
6.24. Simple loops: scalability 76
6.25. Simple loops: speed-up 76
6.26. The two looping mechanism 77
6.27. Domain decomposition into blocks 77
6.28. Looping in blocks 78
6.29. System 3: Performance for different implementations 78
6.30. System 4: Performance for different implementations 79
6.31. Palabos: single core performance 80
6.32. Palabos: memory consumption 81
6.33. Palabos: multi-core performance 81
6.34. Random spheres domain 82
6.35. Random spheres: residence time distribution 85
6.36. Random spheres: mass concentration 86
6.37. Random spheres: Simulation runtime per time step 87

7.1. Realistic porous medium: CT scan 88
7.2. Realistic porous medium: domain creation 88
7.3. Realistic porous medium: simulation domain 89
7.4. Streamlines in realistic porous medium 90
7.5. Realistic porous media: residence time distribution 91
7.6. Realistic porous medium: mass fraction 91

8.1. Realistic porous medium 94

A.1. First law of thermodynamics 98

B.1. Flow regimes 105
B.2. Pressure distribution around an airfoil 107
B.3. Drag and lift force 107

F.1. Possible LBM Lattices 133
F.2. Mesh refinement in LBM 134

G.1. Memory organisation 136
G.2. Multi-level cache 138
G.3. Compilation process in C++ 140
G.4. Scalability in parallel computing 144
G.5. Nested parallelism 145

H.1. Square LDC: pressure 147
H.2. Square LDC: vorticity 148
H.3. Failed simulation: residence time distribution 148

I.1. Gaussian quadrature for rectangular elements 154
I.2. FFT analysis 156

J.1. Harmonic oscillator 159

List of acronyms and nomenclature

1. Acronyms

AD	Advection-diffusion
API	Application programming interface
ASCII	American standard code for information interchange
BB	Bounce-back
BR	Blockage ratio
BGK	Bhatnagar-Gross-Krook
CFD	Computational fluid dynamics
CPU	Central processing unit
CUDA	Compute unified device architecture
D*d*Q*q*	d-dimensional set of q velocities
DNS	Direct numerical simulation
DPD	Dissipative particle dynamics
DSMC	Direct simulation Monte-Carlo
FANS	Favre-averaged Navier-Stokes equations
FDM	Finite difference method
FEM	Finite element method
FHP	Frisch-Hasslacher-Pomeau
FVM	Finite volume method
GCC	GNU compiler collection
GPGPU	General purpose computation on graphics processing unit
GPU	Graphics processing unit
HPP	Hardy-Pomeau-Pazzis
LBM	Lattice-Boltzmann method
LDC	Lid driven cavity
LES	Large eddy simulation
LGA	Lattice gas automata

MD	Molecular dynamics
MEA	Momentum exchange algorithm
micro-CT	Micro–computed tomography
Mlups	Million lattice updates per second
MPC	Multi-particle collision
MPI	Message-passing interface
MRT	Multiple relaxation time
OpenMP	Open multi-processing
OpenMPI	Open message passing interface
NS	Navier-Stokes
OS	Operating system
RAM	Random access memory
RANS	Reynolds-averaged Navier-Stokes equations
RTD	Residence time distribution
SDK	Software development kit
SPH	Smoothed-particle hydrodynamics
TRT	Two relaxation time

2. Nomenclature

2.1. Greek symbols and special characters

$^{(in)}$	Incoming flux	
$^{(out)}$	Outgoing flux	
α	Discrete lattice direction	$[-]$
$\bar{\alpha}$	Reflected population	$[-]$
δ_{ij}	Kronecker delta	$[-]$
δt	Continuous time step	$[s]$
δx	Control volume dimension	$[m]$
δy	Control volume dimension	$[m]$
$\Delta\rho$	Density fluctuations	$\left[\frac{kg}{m^3}\right]$
Δp	Change of momentum	$\left[\frac{kg\,m}{s}\right]$
	Pressure loss	$\left[-\right]$
$\Delta\vec{p}_\alpha$	Change of momentum in lattice direction	$\left[-\right]$

Δt	Discrete lattice time step	$[-]$
Δx	Lattice size	$[-]$
$\Delta \vec{P}$	Change of momentum	$\left[\frac{kg\,m}{s}\right]$
ϵ	Deviation, small parameter	$[-]$
ε	Dissipation rate	$\left[\frac{m^2}{s^3}\right]$
Θ_α	Discrete angle	$[rad]$
κ	Ratio of specific heats	$[-]$
λ	Continuous relaxation time	$[s]$
	Dilatational viscosity	$\left[Pl = \frac{Pa}{s} = \frac{kg}{m\,s}\right]$
λ_m	Mean free path	$[m]$
Λ	Magic parameter for TRT model	$[-]$
μ	Overall dynamic viscosity	$\left[Pl = \frac{Pa}{s} = \frac{kg}{m\,s}\right]$
μ_T	Turbulent dynamic viscosity	$\left[Pl = \frac{Pa}{s} = \frac{kg}{m\,s}\right]$
ν	Overall kinematic viscosity	$\left[\frac{m^2}{s}\right]$
ν_0	Laminar kinematic viscosity	$\left[\frac{m^2}{s}\right]$
ν_d	Dimensioned kinematic viscosity	$\left[\frac{m^2}{s}\right]$
ν_{lb}	LBM kinematic viscosity	$[-]$
ν_T	Turbulent kinematic viscosity	$\left[\frac{m^2}{s}\right]$
$\vec{\xi}, \xi_\alpha$	Microscopic velocities	$\left[\frac{m}{s}\right]$
π	Mathematical constant Pi	3.14
	Hydrostatic stress	$\left[Pa = \frac{N}{m^2} = \frac{kg}{m\,s^2}\right]$
ρ	Density	$\left[\frac{kg}{m^3}\right]$
ρ_∞	Upstream density	$\left[\frac{kg}{m^3}\right]$
σ_{ij}	Stress tensor	$\left[Pa = \frac{N}{m^2} = \frac{kg}{m\,s^2}\right]$
τ	Overall relaxation time	$[-]$
	Tortuosity	$[-]$
	Turnover time	$[s]$
τ_μ	Time in between collisions	$[s]$
τ_0	Laminar relaxation time	$[-]$
τ_{col}	Duration of a collision	$[s]$
τ_i	Relaxation time of component i	$[-]$
τ_T	Turbulent relaxation time	$[-]$
τ, T	Slow times	$[s]$

τ_{ij}	Viscous stresses	$\left[Pa = \frac{N}{m^2} = \frac{kg}{m\,s^2}\right]$
Y_i	Mass fraction	$[-]$
ϕ	Volume fraction, porosity	$[-]$
χ_i	Mole fraction	$[-]$
ψ	Arbitrary function	$[-]$
ω	Collision frequency of BGK model	$[-]$
ω^+, ω^-	Collision frequencies of TRT model	$[-]$
$\dot{\omega}$	Chemical production rate	
Ω	Collision operator	$\left[\frac{kg\,s^3}{m^6}\right]$
$Đ_i^T$	Thermal diffusion coefficient	$\left[\frac{m^2}{s\,K}\right]$
$Đ_{ij}$	Binary diffusion coefficient	$\left[\frac{m^2}{s}\right]$
$\mathcal{H}$	Quantity H, heat	$[-]$

2.2. Roman symbols

a	Thermal diffusivity	$\left[\frac{m^2}{s}\right]$
A	Area	$[m^2]$
	Numerical constant	$[-]$
b	Width of LDC or channel	$[-]$
b_c	Horizontal position of cylinder	$[-]$
B	Numerical constant	$[-]$
c	Lattice speed	$[-]$
	Molar concentration	$\left[\frac{mol}{m^3}\right]$
c_V	Conversion factor for viscosity	$\left[\frac{s}{m^2}\right]$
c_i	Molar concentration of component	$\left[\frac{mol}{m^3}\right]$
c_s	Speed of sound	$\left[\frac{m}{s}\right]$
c_L	Conversion factor for length	$\left[\frac{1}{m}\right]$
c_U	Conversion factor for velocity	$\left[\frac{s}{m}\right]$
C	Numerical constant	$[-]$
C_D	Drag coefficient	$[-]$
C_L	Lift coefficient	$[-]$
C_S	Smagorinsky constant	$0.1 - 0.2$
C_{ijkl}	Elasticity tensor	

d	Diameter of cylinder or sphere	$[-]$
dt'	Differential time step	$[-]$
dx, dy	Differential space element	$[-]$
dA	Differential area element	$[-]$
dV	Differential volume element	$[-]$
D^*_{eff}	Corrected Fickian diffusion coefficient	$\left[\frac{m^2}{s}\right]$
D_i	Fickian diffusion coefficient	$\left[\frac{m^2}{s}\right]$
D^*_i	Dimensionless Fickian diffusion coefficient	$[-]$
e	Specific energy	$\left[\frac{m^2}{s^2}\right]$
	Euler's number	2.72
$\vec{e}_\alpha$	Discrete directions vector	$\left[-\right]$
e_i	Specific internal energy	$\left[\frac{m^2}{s^2}\right]$
E	Efficiency	$[-]$
	Error	$[-]$
$E(t)$	Residence time distribution	$[s]$
E_a	External energy	$\left[J = \frac{kg\,m^2}{s^2}\right]$
E_i	Internal energy	$\left[J = \frac{kg\,m^2}{s^2}\right]$
$E_{\Delta x}$	Error in space	$[-]$
$E_{\Delta t}$	Error in time	$[-]$
E_{Ma}	Compressibility error	$[-]$
f	Distribution function	$\left[\frac{kg\,s^3}{m^6}\right]$
	Frequency	$\left[\frac{1}{s}\right]$
f_α	Discrete distribution function	$[-]$
f^t_α	Temporary discrete distribution after collision	$[-]$
$f^{(eq)}$	Equilibrium distribution	$\left[\frac{kg\,s^3}{m^6}\right]$
$f^{(eq)}_\alpha$	Discrete equilibrium distribution	$[-]$
$f^{(eq)}_{1D}$	1D equilibrium distribution	$\left[\frac{kg\,s^3}{m^6}\right]$
$f^+_\alpha, f^{(eq)+}_\alpha$	Symmetric distribution function	$[-]$
$f^-_\alpha, f^{(eq)-}_\alpha$	Antisymmetric distribution function	$[-]$
$F, \vec{F}$	Force	$\left[N = \frac{kg\,m}{s^2}\right]$
$F(t)$	Cumulative residence time distribution	$[-]$
F_D	Drag force	$\left[N = \frac{kg\,m}{s^2}\right]$
F_L	Lift force	$\left[N = \frac{kg\,m}{s^2}\right]$

g	Difference in momentum	$\left[\frac{kg\,m}{s}\right]$
	gravitational constant	$9.81\,\frac{m}{s^2}$
$g_\alpha, g_{\alpha,i}$	Discrete distribution for AD	$[-]$
$g_\alpha^{(eq)}, g_{\alpha,i}^{(eq)}$	Discrete equilibrium distribution for AD	$[-]$
g_i	Specific force	$\left[\frac{m}{s^2}\right]$
h	Height of LDC, channel or packeted bed	$[-]$
H	Hermite polynomials	$[-]$
h_c	Vertical position of cylinder	$[-]$
i, j, k, l	Cartesian indices[1]	$[-]$
I	Integral	$[-]$
$\vec{j}$	Diffusive mass flux	$\left[\frac{kg}{m^3}\right]$
$J, \vec{J}$	Diffusive molar flux	$\left[\frac{mol}{m^3}\right]$
k	Thermal conductivity	$\left[\frac{W}{m\,K} = \frac{kg\,m}{s^3}\right]$
k_B	Boltzmann constant	$1.38 \cdot 10^{-23}\,\frac{J}{K}$
Kn	Knudsen number	$[-]$
L	Characteristic length	$[m], [-]$
L_d	Dimensioned characteristic length	$[m]$
l_k	Kolmogorov length scale	$[m]$
L_{lb}	LBM characteristic length	$[-]$
l_m	Prandtl mixing length	$[m]$
m	Mass	$[kg]$
	Counter	$[-]$
m_P	Particle mass	$[kg]$
M	Molar mass	$\left[\frac{kg}{mol}\right]$
M_i	Molar mass of component i	$\left[\frac{kg}{mol}\right]$
Ma	Mach number	$[-]$
n	Amount of substance	$[mol]$
	Counter	$[-]$
	index normal to boundary	$[-]$
$\vec{n}$	Normal vector	$[-]$
	Total mass flux	$\left[\frac{kg}{m^3}\right]$
n_i	Amount of component i	$[mol]$

[1]In case of a single summation the index i leads to a new equation whereas j is the corresponding summation index.

N	Number of cells	$[-]$
	Number of molecules	$[-]$
$\vec{N}$	Total molar flux	$\left[\frac{mol}{m^3}\right]$
p	Pressure	$\left[Pa = \frac{kg}{m\,s^2}\right]$
	Number of processors	$[-]$
P	Normalised pressure	$\left[\frac{m^2}{s^2}\right]$
p_0	Static pressure	$\left[Pa = \frac{kg}{m\,s^2}\right]$
p^*	Dimensionless pressure	$[-]$
p_a, p_b	Momenta of particles	$\left[\frac{kg\,m}{s}\right]$
p_d	Dynamic pressure	$\left[Pa = \frac{kg}{m\,s^2}\right]$
p_s	Stagnation pressure	$\left[Pa = \frac{kg}{m\,s^2}\right]$
p_t	Total pressure	$\left[Pa = \frac{kg}{m\,s^2}\right]$
p_∞	Upstream pressure	$\left[Pa = \frac{kg}{m\,s^2}\right]$
Pe	Péclet number	$\left[\,-\,\right]$
q	Specific heat	$\left[\frac{m^2}{s^2}\right]$
Q	Heat	$\left[J = \frac{kg\,m^2}{s^2}\right]$
$\lvert Q\rvert$	Filtered momentum flux	$\left[\frac{kg\,m}{s}\right]$
R	Gas constant	$8.314\,\frac{J}{mol\,K}$
R_m	Specific gas constant	$\left[\frac{J}{mol\,K} = \frac{kg\,m^2}{s^2\,mol\,K}\right]$
Re	Reynolds number	$[-]$
Re_p	Particle Reynolds number	$[-]$
S	Speed-up	$[-]$
	Entropy	$\left[\frac{J}{K} = \frac{kg\,m^2}{s^2\,K}\right]$
	Reference Area	$[m^2]$
S_{ij}	Strain rate tensor	$\left[\frac{1}{s}\right]$
$\lvert S\rvert$	Overall strain	$\left[\frac{1}{s}\right]$
Sc	Schmidt number	$[-]$
St	Strouhal number	$[-]$
t	Time	$[s]$
$\bar{t}$	Average residence time	$[s]$
t^*	Dimensionless time	$[-]$
t_{con}	Convective time scale	$[s]$

t_{dif}	Diffusive time scale	$[s]$
t_{hyd}	Hydrodynamic time scale	$[s]$
t_k	Kolmogorov time scale	$[s]$
T	Temperature	$[K]$
	Runtime	$[s]$
	Characteristic time	$[s]$
	Arbitrary tensor	
$\vec{u}, u_i$	Macroscopic velocity	$\left[\frac{m}{s}\right]$
$\vec{u}^*$	Dimensionless velocity	$[-]$
U	Magnitude of characteristic velocity	$\left[\frac{m}{s}\right]$
U_d	Dimensioned characteristic velocity	$\left[\frac{m}{s}\right]$
u_k	Kolmogorov velocity scale	$\left[\frac{m}{s}\right]$
$\vec{u}^{Y}$	Mass averaged velocity	$\left[\frac{m}{s}\right]$
$\vec{u}^{\chi}$	Molar averaged velocity	$\left[\frac{m}{s}\right]$
$\vec{u}^{\phi}$	Volume averaged velocity	$\left[\frac{m}{s}\right]$
U_{lb}	LBM characteristic velocity	$[-]$
U_∞	Up-stream velocity	$\left[\frac{m}{s}\right]$
$\vec{v}, v_i$	Relative velocity	$\left[\frac{m}{s}\right]$
v	Specific volume	$\left[\frac{m^3}{kg}\right]$
$\vec{v}_p$	Most probable velocity	$\left[\frac{m}{s}\right]$
V_T	Total volume	$[m^3]$
V_V	Void volume	$[m^3]$
$\dot{V}$	Volume flux	$\left[\frac{m^3}{s}\right]$
$\vec{w}_c$	Correction velocity	$\left[\frac{m}{s}\right]$
$\vec{w}_i$	Diffusion velocity	$\left[\frac{m}{s}\right]$
W	Work	$\left[J = \frac{kg\,m^2}{s^2}\right]$
W_α, w_α	Discrete weights	$[-]$
W_V	Volume work	$\left[J = \frac{kg\,m^2}{s^2}\right]$
$\vec{x}, x, x_i$	Parameter for position	$[m]$
x_i^w	Boundary link coordinates	$[-]$
$\vec{x}^*$	Dimensionless parameter for position	$[-]$
y, z	Parameter for position	$[m]$

3. Figures and diagrams

In the context of this thesis, for the sake of clarity, also a certain *consistent visual language* is chosen. The following sections explains how the corresponding illustrations should be interpreted. The basic vector images were all created in LaTeX using Ti*k*Z and PGFPlots while the 3*D* renderings were created with Paraview, some applied Laplacian smoothing, exported to Blender as *.x3d-files and rendered with the render engine Cycles.

3.1. Curves and diagrams

Curves not created using the self-written C++ code are always coloured grey in order to emphasise that they are only mentioned for reference. This applies to data taken from other publications but also to different LBM implementations such as Palabos. The corresponding marker for discrete data points is in any case a solid circle.
Curves derived using the C++ code on the other hand are always coloured black. For performance benchmarks where two data sets are present for the same markers the lower one always corresponds to results obtained with a single core and the upper one corresponds to the full usage of an entire processor with six cores. The Matlab code is completely vectorised and therefore Matlab will try to use all cores wherever possible. The data markers are consistent throughout the thesis and are chosen as follows: Matlab BGK ●, C++ BGK ○, C++ TRT $\times$, C++ BGK-LES □ and C++ TRT-LES $+$ where LES (large-eddy simulation) denotes the corresponding collision operators with a Smagorinsky turbulence models.

3.2. Discrete distribution functions

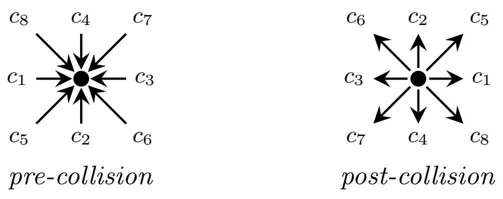


Figure 1.: Graphic representation of pre-collision (left) and post-collision populations (right)

Lattice-Boltzmann can be seen as *clusters of particles streaming and colliding on a discrete grid*, the lattice. This view is also supported visually: Populations before the collision step point towards the rest node, represented by the centre dot, while populations after the collision step point away from it (figure 1). Note that the populations do not change direction due to collisions - they are only rescaled: A population will not change its direction but will change in amount. For the sake of simplicity populations will be pictured always filling a cell entirely but should be imagined as arrows with different length that reflect the amount of the distribution function and therefore are a measure for the macroscopic state. For a real lattice the rest node would be the biggest population and the diagonal populations would be comparably small. In order to distinguish bulk fluid from boundary nodes additionally two different colours for solid and fluid nodes are adopted (figure 2). Finally populations not relevant for an explanation are coloured in light gray whereas the relevant populations are coloured in black.

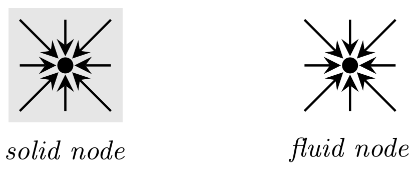


Figure 2.: Graphic representation of solid (left) and fluid nodes (right)

"Ars longa,
vita brevis,
occasio praeceps,
experimentum periculosum,
iudicium difficile."

Dedicated to my beloved sister Ruth.

Preface

Even though I tried to put the focus of my master programme on fluid dynamics, I can't remember hearing about the alternative kinetic models of Lattice-Boltzmann. All the more was I impressed not only by the simplicity and computational speed of such methods but even more by the theory behind it. I think it perfectly illustrates that every model, every physical description, in the end is just an *abstraction*, an *approximation of an intangible, complex nature*, but nevertheless - even though just a rudimentary attempt to predict the future, to find rules in all the chaos - no less *fascinating*.
As the topic touches several different areas of study I spent most time reading books and publications. To somebody new to the field I would advise to not bother so much about scientific papers and rather getting familiar with the basics of fluid dynamics and molecular gas dynamics first: The German-language book *"Molekulare Gasdynamik"* written by *D. Hänel* offers an excellent introduction into the kinetic theory of gases as well as a small insight into the Lattice-Boltzmann methods. In order to get an extensive overview of all the basics in Lattice-Boltzmann I can recommend *"The Lattice-Boltzmann Method: Principles and Practice"* by *Krüger et al.*: It covers the entire framework from theory to practice and includes coding samples that can also be accessed on-line. If somebody is further interested in the detailed derivation of the discrete algorithm I would pick up the original papers written by *He et al.*. Finally, for the section on multi-component flow, I suggest *"Diffusion: Mass Transfer in Fluid Systems"* by *E.L. Cussler* as it gets along without rigorous mathematical formulations.
As getting familiar with the topic soon amounts to a lot of literature research in a lot of different fields I tried to summarise the most important concepts in this thesis and included a lot of basics and interesting connections I came across in the appendix that should make this thesis also readable for somebody from another field with a basic understanding of maths. While I tried to work as scientifically as possible in the thesis itself, the appendix does not raise the requirement to be complete and is therefore less underpinned with citations: It is a loose collections of ideas with a smooth transition between things I have picked up over the years and my own ideas.
Finally I would like to sincerely thank my supervisor *Renè* for his constant support and patience, that allowed me to work at my pace, and to *Markus* for several interesting talks and for supplying me with an excellent Fluent simulation to validate against.
Last but not least I would like to say thank you to my parents, *Tobias* and *Erika*, that have always supported me - not merely financially - throughout my studies. It is only now that I realise that most of my skill set is nothing more than a mere product of my dad's discipline, his strictness and mathematical precision and my mum's idealism, her inexhaustible patience, her passionate dedication and her eye for the detail. Furthermore I am also very grateful for my two brothers, *Jonas* and *Elias*, both with manifold interests, that constantly encourage me to learn new things. In particular I would like to dedicate this thesis to my sister *Ruth* that sadly is no longer with us: With her the world has not only lost a brilliant young scientist but even more a sensitive and genuinely loving human being.

"Modern kinetic theory offers a unifying theoretical framework within which a great variety of seemingly unrelated physical systems that exhibit complex dynamical behaviour can be explored in a coherent manner."

- John Karkheck

Motivation

Over the last couple of years, with the rise of powerful computers, the field of computational physics has progressively gained importance and partially replaced expensive and time consuming experiments. This is especially true for the sector of *fluid dynamics* as the *non-linear coupled partial differential equations* describing fluid flow can only be solved analytically for a few simplified cases. The science of obtaining corresponding *approximate* computer-based solutions is called *computational fluid dynamics* (CFD).
The field of computational fluid dynamics is dominated by the *finite volume method* (FVM) that is based on the direct discretisation of the conservation equations of fluid dynamics on structured or unstructured meshes. Meshing and simulating highly complex geometries poses though several challenges in terms of resolution and computational speed and therefore the simulation of intricate porous media is still subject of ongoing research.
In order to lift those restrictions different particle based methods have emerged. One family of methods, namely the *Lattice-Boltzmann methods* (LBM), is based on a kinetic approach and shows significant advantages in particular for incompressible flow in intricate geometries with high temporal and spatial resolution. In this thesis a corresponding *high performance C++ code*, running in *parallel* on all cores of a single processor, is implemented, *benchmarked* in terms of computational speed against another novel implementation and physically *verified* using standard benchmark scenarios and an Ansys Fluent simulation. The stability of two common collision operators, *BGK*, a simple collision operator with one relaxation time, and *TRT*, a relaxation operator with an additional, second, freely tunable relaxation time, is investigated, the effective range regarding the Reynolds number then extended using a *large-eddy turbulence model* and a *second lattice*, accounting for the *advection and diffusion of a species*, is included. Finally this model is applied to *flow in complex porous media*.
The first chapter outlines the basics of computational fluid dynamics, the governing equations, traditional methods and their drawbacks. The second chapter introduces the Lattice-Boltzmann methods from a standpoint of the kinetic theory of gases and explains the basics of this algorithm, the accuracy, stability, initial and boundary conditions. In the third chapter the physics of multi-component flows are explained while in the fourth it is demonstrated how this model can be included into a Lattice-Boltzmann simulation. The next chapter describes the characteristics of flows in porous media. In the sixth chapter the particular implementation is discussed and benchmarked against standard benchmark scenarios for single and binary multi-component flow and in the final seventh chapter this verified model is applied to binary flow in realistic complex porous media obtained by a tomography scan.

1. A brief introduction to computational fluid dynamics

The following chapter gives a short introduction into the basics of fluid mechanics and computational fluid dynamics (CFD) including the governing equations, traditional methods and their drawbacks. The concepts presented can be found in every good reference book for fluid dynamics.

1.1. Fluid mechanics

In fluid mechanics the *macroscopic properties* of a fluid are of interest and hence the fluid is approximated by a continuum: The mechanical behaviour is modelled through *continuous blobs of mass that completely fill space* rather than individual particles.[2] The smallest brick of this model, a single fluid element, is small with respect to the system size but large compared to the size of the molecules and the distance between them. This means the corresponding limit values for density, stresses and specific forces

$$\rho := \lim_{\Delta V \to 0} \frac{\Delta m}{\Delta V} \qquad \sigma_{ij} := \lim_{\Delta A_i \to 0} \frac{\Delta F_j}{\Delta A_i} \qquad g_i := \lim_{\Delta m \to 0} \frac{\Delta G_i}{\Delta m}$$

must exist.[3]

1.1.1. Conservation equations for compressible flow

In continuum based fluid dynamics there are three different governing coupled differential equations that describe the *conservation of mass, momentum and energy* on the macroscopic level and have to be solved in computational fluid dynamics. They can be derived by looking at a fluid parcel and analysing the changes that arise from flow over the surface or sources inside it. The resulting equations can be formulated either in *differential* (table 1.1) or *integral notation* (table B.1 in appendix B) and can be transformed into each other using Gauss's divergence theorem (appendix I.3.2).

[2]The science analysing systems described by these so called continua is called *continuum mechanics* and is widely used throughout physics mainly in solid and fluid mechanics. For a consistent description of such a system vectors should be distinguished regarding their transformation behaviour into co- and contravariant vectors. As the deformations in this context can be assumed small this may be neglected.

[3]The number of molecules must be large enough to compensate every stochastic fluctuation in the flow field. For most applications in fluid dynamics this is a sufficient approximation but it breaks down for the extremes of dilute gases as well as on microscopic level and other approaches like the kinetic theory of gases (section 2.1.1) must be applied.

$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0$	*Continuity equation* conservation of mass
$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + \rho g_i$	*Momentum equation* (Navier-Stokes equation) conservation of momentum
$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho u_j e)}{\partial x_j} = \frac{\partial q_j}{\partial x_j} + \frac{\partial(\sigma_{ji} u_i)}{\partial x_j} + \rho u_j g_j$	*Energy equation* conservation of energy

Table 1.1.: The conservation equations of fluid mechanics in differential notation

Where the specific total energy e is the sum of internal and macroscopic energy[4]

$$e = e_i + \frac{u_k u_k}{2}$$

and the local heat flux density is given by *Fourier's law* of heat conduction[5]

$$q_j = -k \frac{\partial T}{\partial x_j}.$$

1.1.1.1. Deformations and stress tensor: Stokes' law

In *theory of linear elasticity* it is assumed that the stress tensor σ_{ij} can be calculated using the elasticity tensor C_{ijkl}[6] and the rate-of-strain tensor S_{kl} according to

$$\sigma_{ij} = \sigma_{ij}^{(0)} + C_{ijkl} S_{kl}.$$

where $\sigma_{ij}^{(0)}$ is the stress distribution in the resting state of the continuum of interest. Latter corresponds to the hydrostatic pressure p_0 in a resting and to the thermodynamic pressure p in a moving fluid[7]

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij}.$$

For an *isotropic material* where there is no preferred direction and further assuming a symmetric Cauchy stress tensor $\tau_{ij} = \tau_{ji}$, the elasticity tensor degenerates to

$$C_{ijkl} = \lambda \delta_{ijkl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}),$$

leaving us with only two independent coefficients μ (*shear viscosity*) and λ (dilatational viscosity). The corresponding viscous stresses τ_{ij} can be calculated assuming a linearly proportionality to the rate of change of the fluid's velocity, the shear rate, (*Newtonian fluid*) to[8]

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda \frac{\partial u_k}{\partial x_k} \delta_{ij}.$$

[4] A small introduction to thermodynamics can be found in appendix A.

[5] Transport laws assume the proportionality of the gradient and a corresponding transport coefficient. Although found empirically, they can be derived using the later mentioned kinetic theory of gases.[1]

[6] A fourth rank tensor with $3^4 = 81$ coefficients in its most general form.

[7] Here we apply Pascal's law: Pressure acts equally on all directions of a particular point in space. The stresses in a fluid at rest are isotropic and no shear stresses are present.

[8] For *non-Newtonian fluids*, where the flow properties are different from the aforementioned Newtonian fluid (e.g. the viscosity is a function of the shear-rate), other approaches have to be used. Even though almost all fluids show more or less non-Newtonian behaviour, most, like water and air, can be approximated as Newtonian.

Introducing the mean mechanical pressure $\overline{p}$,[9]

$$\overline{p} := -\frac{1}{3}\left(\sigma_{11}+\sigma_{22}+\sigma_{33}\right) = p - \left(\lambda + \frac{2}{3}\mu\right)\frac{\partial u_k}{\partial x_k}$$

leads to a surprising result: Unless either the divergence of the velocity or the term $\lambda + \frac{2}{3}\mu$, often referred to as *bulk viscosity*, are zero, the mechanical pressure is not equivalent to the thermodynamic pressure.[10] [2]
Stokes simply assumed a vanishing bulk viscosity (Stokes' hypothesis) [11] which left him with a stress tensor according to

$$\sigma_{ij} = -p\delta_{ij} - \frac{2}{3}\mu S_{kk}\delta_{ij} + 2\mu S_{ij}$$

where

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

Three possible basic deformations of a continuum element can be identified: angular and linear deformation as well as volumetric dilatation (figure 1.1). While the latter is caused by the first two terms in the stress tensor the last term results in angular deformation for $i \neq j$ and linear deformation for $i = j$.

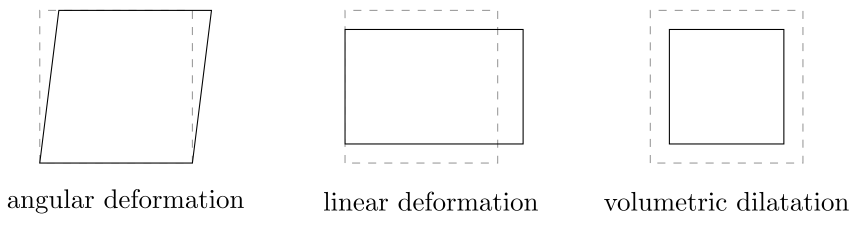


Figure 1.1.: Possible deformations of a continuum element

1.1.1.2. Equation of state

Finally in order to close the set of equations an *equation of state* $p = p(\rho, e_i)$ has to be considered. For an ideal gas this is given by the ideal gas law

$$pV = nRT.$$

Else other approaches have to be taken like the van der Waals equation and the virial expansion for real gases or the Tait equation for liquids.

[9]Analogously to the hydrostatic stress in linear elasticity theory $\pi = \frac{\sigma_{kk}}{3}$.
[10]The bulk viscosity introduces additional dissipation during a change of volume, where shear forces are not present.
[11]Buresti [3] suggests this should be seen as $\overline{p} \approx p$. Even though this assumption is widely used throughout fluid dynamics it seems as if it would only hold for mono-atomic gases, while in the case of poly-atomic gases molecular interactions are probably responsible for a thermodynamic pressure that deviates from the mechanical pressure.

1.1.2. Speed of sound and Mach number

Looking closer at the integral conservation equations[12] one can identify several possible discontinuities with abrupt changes of the flow field, given by the *Rankine-Hugoniot jump conditions*, that satisfy the conservation equations. One of them is a *normal shock* where velocity, density and pressure change. We can investigate this particular solution further by looking at a simplified one dimensional stationary flow with constant area using the *stream filament theory*. In the limit of small perturbations, that can be assumed isentropic, one can neglect terms of higher order leaving us with a model like shown in figure 1.2. The propagation of this small perturbation is then characterised by the equations in table 1.2.

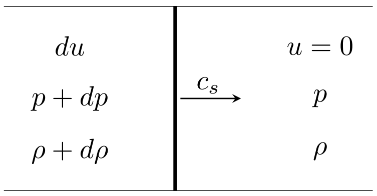


Figure 1.2.: Propagation of a small perturbation with changes in velocity, pressure and density

$\rho du = c_s d\rho$	*Continuity equation*
$dp - 2\,\rho\, c_s\, du + c_s^2 d\rho = 0$	*Momentum equation*

Table 1.2.: Conservation equations for the propagation of a small perturbation

Combining the continuity and the momentum equation leads to[13]

$$c_s^2 = \left(\frac{\partial p}{\partial \rho}\right)_{S=const}.$$

This characteristic velocity, referred to as the *speed of sound*, is the *speed of propagation of small perturbations* for the case of compressible fluids.
In case of an ideal gas for isentropic changes of state further $\frac{p}{\rho^\kappa} = const$ holds (appendix A.6), which leads to a speed of sound that is given by

$$c_s = \sqrt{\kappa\, R_m\, T}$$

where $R_m = R/M$.
In fluid dynamics generally *dimensionless numbers* are introduced to compare the behaviour of different similar flows[14]: Fluid flows sharing the same relevant dimensionless numbers exhibit the same physical behaviour.[15]

[12]These discontinuities are not present in the differential notation as we have to assume continuously differentiable variables for the derivation.

[13]Similarly to this adiabatic speed of sound for isothermal flow one might introduce the isothermal speed of sound $c_s = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)_T}$ which for an ideal gas yields $c_s = \sqrt{R_m\, T}$.

[14]For example similar problems of different scales

[15]*Law of similarity*: For different flows though all dimensionless numbers can never be identical.

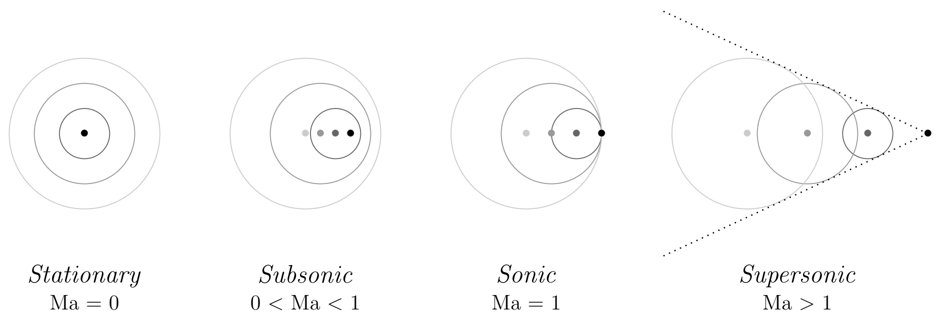


Figure 1.3.: Flow velocity and the speed of sound at different Mach numbers

One of the most important, the *Mach number*, represents the ratio of the flow velocity past an object U and the local speed of sound c_s.

$$Ma = \frac{U}{c_s} \qquad \frac{\text{ordered kinetic energy}}{\text{random kinetic energy}}$$

When an object is moving through a fluid, it pushes fluid particles in front of it away, creating pressure disturbances in form of waves spreading at the speed of sound, so called Mach waves. As can be seen in figure 1.3 this leads to a *completely different propagation information* depending on the Mach number: With increasing velocity of the travelling object information spreads progressively non-uniform in space.

1.1.2.1. Non-dimensional conservation equations

Additional dimensionless numbers can be found by non-dimensionalising the governing differential equations, introducing the *characteristic measures*

$$x_i^* = \tfrac{x_i}{L} \qquad u_i^* = \tfrac{u_i}{U} \qquad \rho^* = \tfrac{\rho}{\rho_0} \qquad T^* = \tfrac{\Delta T}{\Delta T_0}$$

$$g_i^* = \tfrac{g_i}{g} \qquad t^* = \tfrac{t}{\frac{L}{U}} \qquad p^* = \tfrac{p}{\rho_0 U^2}$$

Under the assumption of a perfect gas (appendix A.4)[16] this leads to

$$\frac{\partial \rho^*}{\partial t^*} + \frac{\partial(\rho^* u_i^*)}{\partial x_i^*} = 0$$

$$\frac{\partial(\rho^* u_i^*)}{\partial t^*} + \frac{\partial(\rho^* u_j^* u_i^*)}{\partial x_j^*} = -\frac{\partial p^*}{\partial x_i^*} + \frac{1}{Re}\frac{\partial \tau_{ij}^*}{\partial x_j^*} + \frac{1}{Fr^2} g_i^*$$

$$\rho^* \frac{\partial T^*}{\partial t^*} + \rho^* u_j^* \frac{\partial T^*}{\partial x_j^*} = Ec\left(\frac{\partial p^*}{\partial t^*} + u_j^* \frac{\partial p^*}{\partial x_j^*}\right) + \frac{1}{PrRe}\frac{\partial}{\partial x_j^*}\left(\frac{\partial T^*}{\partial x_j^*}\right) + \frac{Ec}{Re}\frac{\partial}{\partial x_j^*}\left(\tau_{ij}^* u_i^*\right)$$

Table 1.3.: The non-dimensional conservation equations in differential notation

where the relevant dimensionless numbers are given by

[16]The energy equation is written in non-conservative notation (appendix B.3).

$Re := \frac{UL}{\nu}$	$\frac{\text{inertial forces}}{\text{viscous forces}}$	*Reynolds number*
$Ec := \frac{U^2}{c_P \Delta T_0}$	$\frac{\text{heat dissipation potential}}{\text{advective transport}}$	*Eckert number*

As can be seen from the non-dimensional energy equation[17] the temperature is decoupled from the equation system if the Eckert number can be assumed comparably small which is the case for small Mach numbers as the relation

$$Ec = \frac{U^2}{c_P \Delta T_0} \frac{c_s^2}{c_s^2} = Ma^2 (\kappa - 1) \frac{T_0}{\Delta T_0}.$$

holds.

1.1.3. Conservation equations for incompressible flow

For special cases the conservation equations can be further simplified.[18] If the density within a fluid parcel can be assumed constant along its streamline (*incompressible flow*), $\nabla \cdot \vec{u} = 0$[19] holds, leading to a favourable yet relevant form (table 1.4).

$\frac{\partial u_j}{\partial x_j} = 0$	*Divergence free condition*
$\frac{\partial \rho}{\partial t} + u_j \frac{\partial \rho}{\partial x_j} = 0$	*Density equation*
$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \frac{1}{\rho} \frac{\partial}{\partial x_j} \left(\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right)$	*Momentum equation*

Table 1.4.: The conservation equations for incompressible flow

If further the density of the initial configuration is assumed constant (*incompressible fluid*), $\nabla \cdot \rho = 0$ holds and the density equation simplifies to $\frac{\partial \rho}{\partial t} = 0$.[20] Generally also the shear viscosity μ is assumed constant leading to the equation system in table 1.5.

$\frac{\partial u_j}{\partial x_j} = 0$	*Continuity equation*
$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\nu \frac{\partial u_i}{\partial x_j} \right)$	*Momentum equation*

Table 1.5.: The conservation equations for incompressible fluids with constant material values

There is no equation of state anymore: The *energy equation* is *decoupled* from the equation system. This means continuity and momentum equation can be solved independently and the temperature can then be calculated from the resulting velocity and density

[17]The definition of the other two dimensionless variables can be found in appendix B.4.
[18]In this section also mass forces are neglected.
[19]This is equivalent to a vanishing volumetric dilatation (appendix B.6.1).
[20]This is equivalent to a constant density throughout the flow field

fields after each time step if necessary. From the conservation equation one is able to recover the pressure gradients but in order to get its absolute value a trick has to be used to couple velocity and density again, resulting in the *Poisson's equation* for pressure[21]

$$\frac{1}{\rho}\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial u_j}{\partial x_i}\frac{\partial u_i}{\partial x_j}.$$

1.1.3.1. Low Mach number approximation

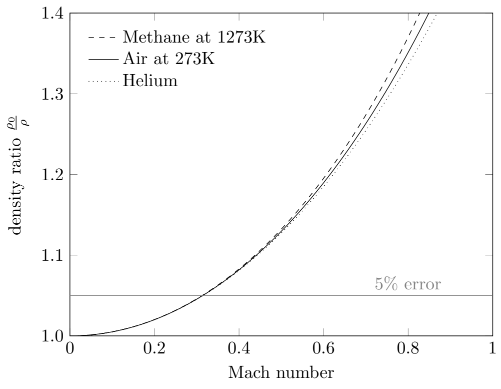


Figure 1.4.: Density ratio over Mach number for different gases

For compressible perfect gases the density ratio (figure 1.4)[22] can be calculated to

$$\frac{\rho_0}{\rho} = \left(1 + \frac{\kappa - 1}{2} Ma^2\right)^{\frac{1}{\kappa-1}}.$$

As can be seen every flow is somewhat compressible but if the Mach number is less than 0.3 the density changes due to pressure are less than 5% and the flow can be approximated with good accuracy as incompressible. [4] A truly incompressible fluid has an *indefinitely large speed of sound*:[23] It instantly knows about small changes in the flow field.[24]

1.1.4. Turbulent flow and Kolmogorov microscales

Flows with low Reynolds number are characterised by only slowly varying velocity fields in which flow happens in layers that float on top of each other without creating cross currents and are therefore referred to as *laminar flows*. The fluid viscosity is high enough to dampen instabilities. As the Reynold number increases the flow departs from smooth flow. This so called *turbulent flow* is characterized by chaotic, nearly random fluctuations in velocity and pressure: The vorticity is non-zero and is

[21]For the derivation see appendix B.6.2.

[22]The derivation can be found in appendix B.7.

[23]Due to $Ma = \frac{U}{c_s} \ll 1$ one can assume that $c_s^2 = \left(\frac{\partial p}{\partial \rho}\right)_S \to \infty$ and therefore $\left(\frac{\partial \rho}{\partial p}\right)_S \to 0$. For this reason in the literature often $\rho \neq \rho(p)$ is found as a definition of incompressibility.

[24]The differential equations are now elliptic instead of hyperbolic.

distributed among vortices of different sizes (figure 1.5).[25] According to *Kolmogorov's hypothesis* energy is supplied at a macroscopic level leading to unstable eddies that break up and gradually pass on the energy ϵ to smaller eddies until on the smallest scale, the so called *Kolmogorov scale* l_k, the energy is dissipated by viscosity.[26] While the large eddies still contain information about the geometry this information gets lost along this energy cascade and the small scale eddies might be assumed universal, homogeneous and isotropic.

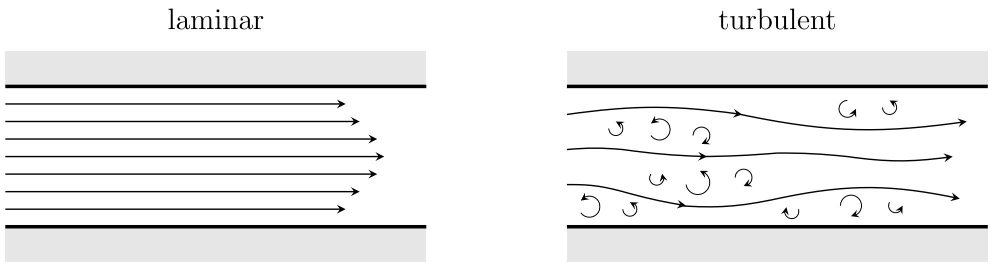


Figure 1.5.: Schematic laminar (left) and turbulent flow (right)

The energy supplied to the fluid on macroscopic scale can be estimated under these assumptions by dimension analysis using the kinetic energy $\propto U^2$ and the corresponding time scale $\propto L/U$ to

$$\varepsilon \sim \frac{U^3}{L}.$$

Whereas the smallest length can be calculated using the dissipation rate ε and the viscosity ν

$$l_k \sim \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}}.$$

This can be used to evaluate the ratio between the largest and smallest scale involved in turbulent fluid flow using the aforementioned Reynolds number

$$\frac{L}{l_k} \sim \left(\frac{U\,L}{\nu}\right)^{\frac{3}{4}} = Re^{\frac{3}{4}}.$$

Same thing can be applied to the corresponding time and velocity scales leading to[27]

$$\frac{T}{t_k} = \sqrt{Re}, \qquad \frac{U}{u_k} = Re^{\frac{1}{4}}.$$

1.1.4.1. Boussinesq hypothesis and Prandtl mixing-length

Several efforts were made in order to predict the effects of turbulence. One popular way is to *filter* the governing equations and focus on the large-scale turbulence while

[25]Turbulent flow is always three-dimensional and transient.

[26]While dissipation during this turbulent energy cascade on scales $l > l_k$ can be neglected.

[27]This means for higher Reynolds numbers these turbulent fluctuations become more fine scale and accordingly the discretisation needed to capture these phenomena in CFD. As can be seen the number of cells scales with $\mathcal{O}(Re^{\frac{9}{4}})$ for three-dimensional turbulent flow.

only modelling the small scale uniform turbulence. The first such models, introduced by *Boussinesq*, proposes an additional *turbulent viscosity* caused by turbulent stresses. *Prandtl* further assumed that Newton's law for viscous fluids still holds but an additional turbulent viscosity μ_T has to be considered. Unlike its counterpart μ this viscosity is not constant in space, it is a property of the flow rather than of the fluid itself. The corresponding kinematic viscosity ν_T is modelled again as a function of the velocity shear and a corresponding mixing length l_m that accounts for a different eddy viscosity depending on the local distance from the wall

$$\nu_T = \frac{\mu_T}{\rho} = l_m^2 \left| \frac{du}{dz} \right| .$$

1.2. Computational fluid dynamics

$$\begin{pmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \cdot & \\ & & \cdot & \cdot & c_{n-1} \\ 0 & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_n \end{pmatrix}$$

Figure 1.6.: In CFD equation systems involving sparse, diagonally dominant matrices, preferably of tri-diagonal form like above, have to be solved in order to obtain a solution. Generally this is done iteratively using the Gauss-Seidel method.

An exact analytical solution to the governing conservation equations can only be found for very few special simplified cases. In order to gain at least an approximate solution in conventional CFD the relevant differential equations are *discretised*. This requires also the computational domain to be represented by a *finite set of points or cells* leading to an approximated and simplified geometry of the real system. For every cell the differential equation can be rewritten as a balance of fluxes from neighbouring cells and every boundary where information is known about the state of the fluid has to be replaced with equations resembling these *boundary conditions*: Instead of the differential equations being solved analytically a linear equation system has to be solved *numerically and iteratively* (figure 1.6).

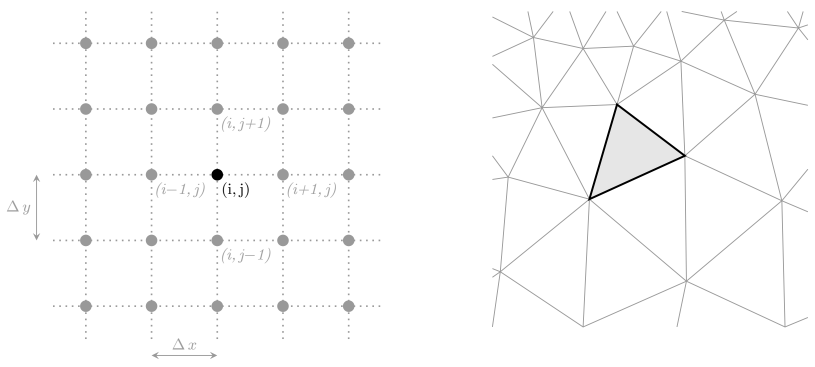


Figure 1.7.: Typical grids for finite difference (left) and finite volume method (right)

In the *finite difference method* (FDM) this is done by expanding the differential notation of the conservation equations in a Taylor series approximating differentials by differences between nodes. This leads to a comparably fast algorithm while though losing conservativity and limiting the use mainly to regular grids. In the *finite volume method* (FVM) the integral notation is discretised and the domain is approximated by cells. The center point of each cell represents the average measures of that particular cell that change due to fluxes from neighbouring cells leading to algebraic equations (figure 1.7). FVM are generally slower but more accurate and can be used with unstructured meshes making up for the slightly slower performance. Therefore modern commercial fluid solvers are generally finite volume based.
Furthermore there are two different kind of approaches for solving the governing differential equations, a *pressure-based*, historically used for incompressible flow, and a *density-based* one, initially designed for compressible flow. In both cases the velocity is obtained from the momentum equation but in the density-based approach the density field is determined from the continuity equation and the pressure is then calculated using the equation of state. As seen before for incompressible flow density and pressure are not directly coupled by an equation of state any more: Generally using the values from the last iteration uncorrected fluxes have to be calculated. Then a pressure correction equation, resulting from a smart combination of continuity and momentum equation, has to be solved and the velocity field has to be adjusted accordingly.
Approximating the differential equations leads to the so called *truncation error*. It is estimated after every step by asymptotic analysis, scaled and outputted as a relative measure for convergence (*residuals*). It should get smaller as a finer mesh is chosen (*grid convergence*) and with similar reasonable meshes the results should also have comparable results (*grid independence*). These two characteristics have to be proven for every simulated system in order to guarantee a correct simulation.

1.2.1. Boundary conditions

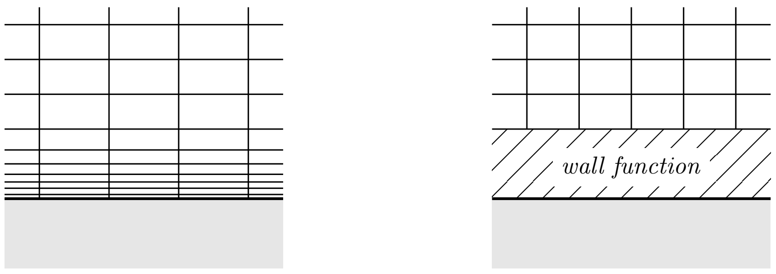


Figure 1.8.: Near wall grid without (left) and with wall function (right)

Boundary conditions are crucial for numeric accuracy: A single incorrect boundary may be streamed into the flow field and affect the whole domain.[28] Additionally the user has to pay attention to not under- or over-specify a particular problem.[29]
Solid walls slow down the fluid locally: The velocity steadily decreases and takes the

[28]Generally speaking large gradients near an in- or outlet are a sign for incorrect problem specification.
[29]In an underspecified problem there are too little boundary conditions and the problem is incompletely specified. In an over-specified problem on the other hand there are more boundary conditions given than can be realised.

value zero at the wall leading to large gradients that have to be resolved. The *near wall sub layer* where the viscous effects play an important role has to be resolved adequately with a relatively fine mesh leading to a longer simulation runtime or be modelled by a wall function taking into account the effects of the boundary layer empirically but reducing accuracy (figure 1.8).

1.2.2. Turbulence modelling

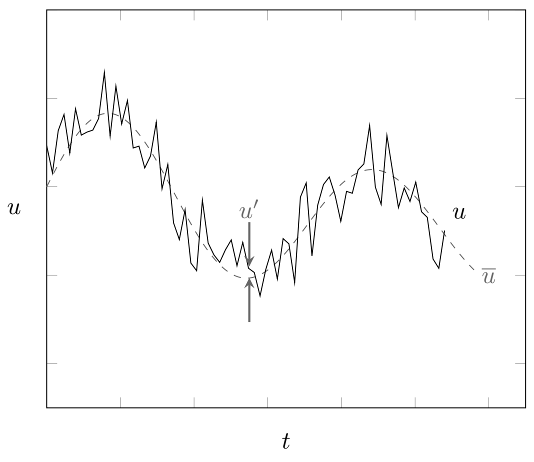


Figure 1.9.: RANS: Turbulent flow with velocity u is modelled as superimposed streaming with velocity $\overline{u}$ and a fluctuating motion with u'.

Most flows for relevant technical applications are highly turbulent ($Re > 10^5$): The flow includes eddies on a smaller scale and a finer mesh as well as very small time steps would be needed in order to resolve the flow down to the Kolmogorov length scale and model the phenomena correctly. This is computationally very expensive but still done in the so called *direct numerical simulation* (DNS) which is mainly used for the low Reynolds number regime and for research purposes.
As this is generally not viable, significant effort has been put into developing models that are less computationally heavy but still maintain accuracy. This is achieved by either statistical or spatial averaging. *Large Eddy Simulations* (LES) apply spatial filtering: Large turbulence that depends on the geometry of the system is resolved while uniform small scale turbulence is modelled. This means one should know beforehand which scales have to be resolved and which should be modelled. Even though less computationally heavy than DNS it is still mostly limited to supercomputers.
Generally another approach is taken by employing stochastic tools and decomposing velocity and pressure into two parts, a time-averaged (defined by the root mean square) and a fluctuating turbulent part (figure 1.9), leading to the *Reynolds-averaged Navier-Stokes equations* (RANS, generally used for incompressible fluids) or *Favre-averaged Navier-Stokes equations* (generally used for compressible fluids) depending on whether an average or density weighted average is used. Due to the non-linearity[30] this leads to an additional term, the so called Reynolds stress term $\tau'_{ij} = \rho\overline{u'_i u'_j}$ (table 1.6), that has to be modelled: There are several so called *turbulence models* available that try to

[30]Precisely due to $\overline{u_i u_j} = \overline{u}_i\,\overline{u}_j + \overline{u'_i u'_j}$

close the equation system by introducing constraints ranging from simple algebraic equations to several additional differential equations describing the transport of those stresses.[31]

$p = \overline{p} + p', \;\; u_i = \overline{u}_i + u'_i$	*Reynolds decomposition*
$\frac{\partial \overline{u}_i}{\partial x_i} = 0$	*Continuity equation*
$\frac{\partial \overline{u}_i}{\partial t} + \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{1}{\rho}\frac{\partial}{\partial x_j}(\tau_{ij} - \rho\overline{u'_i u'_j})$	*Momentum equation*

Table 1.6.: Reynolds-averaged Navier-Stokes equations for incompressible fluids

1.2.2.1. Smagorinsky-Lilly turbulence model

In order to not resolve every scale down to the Kolgomorov length LES introduces spatial filtering of smaller scales (generally smaller than the grid resolution) and describing the effects on those unresolved scales with sub-grid turbulence models. A common approach is to link *turbulence* on this microscopical level to an *increase in viscosity* acting like a low-pass filter, damping short-wavelength oscillations. This turbulent viscosity has to be modelled adequately.
Smagorinsky was the first to propose a formula for CFD, based on an eddy viscosity in 1963. In 2D and 3D problems the velocity shear is an entire tensor given by the local rate-of-strain tensor S_{ij}. The overall strain is then given by its norm

$$|S| = \sqrt{2 S_{ij} S_{ij}}$$

and replaces the velocity shear in Prandtl's mixing-length concept such that

$$\nu_T = l_m^2 |S|.$$

The mixing length is replaced by a product of a constant, the so called *Smagorinsky constant*[32] $C_S = 0.1 - 0.2$ and the filter width Δ, which is generally the grid size,

$$\nu_T = (C_S\,\Delta)^2\,|S|.$$

1.2.3. Insufficiencies of conventional methods

Although CFD has become a standard, complex flow simulations are still challenging and error prone mainly depending on the user's expertise. Generically traditional CFD methods based on a direct discretisation of the Navier-Stokes equations have a couple of limitations: With their macroscopic nature modelling *microscopic phenomena* such as *reactive, multicomponent* as well as *multiphase flows* can be challenging. Normally this

[31]Setting the relevant turbulence model parameters is though very challenging and their influence on the simulation is hard to predict.

[32]In order to take into account the near wall effects more complex models with a dynamic Smagorinsky constant or applying additional a-priori knowledge of the boundary layer, like the van-Driest dampening function, have emerged.

involves including additional differential equations that have to be solved making it computationally expensive. Modelling *free surfaces* where the solution region changes as the surfaces moves and in turn the motion of the surface is determined by the solution may pose difficulties. In the case of *rarefied gases* the underlying continuum approach loses its validity and Navier-Stokes based methods are not valid any more. Additionally, in particular for complex geometries, most time is spent generating adequate meshes[33] for simulations rather than simulating. Probably the biggest drawback though stems from the nature of the differential equations being solved rendering an efficient *parallel implementation* challenging.

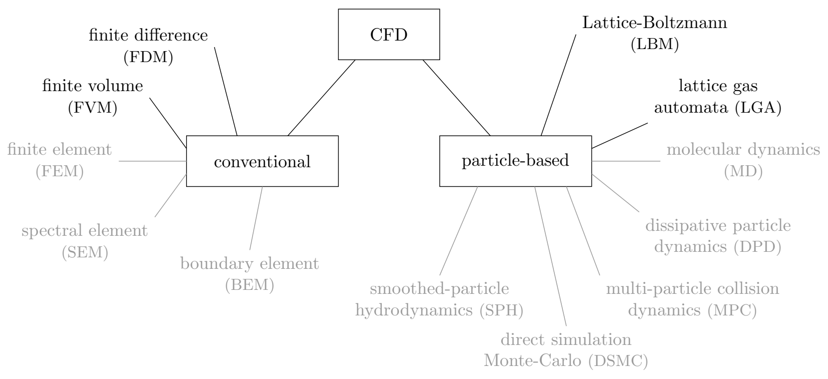


Figure 1.10.: One possible classification of CFD methods as proposed by Krüger et al. [5]

Over the years multiple *particle-based methods* have emerged in order to lift at least some of those limitations. In this case the fluid is represented by discrete particles in the form of single atoms, molecules or artificial clusters of molecules instead of a continuum that interact on a short range with each other. The one that probably got the most attention due to its flexibility are the *Lattice-Boltzmann methods* that will be investigated further in this thesis. One possible classification of CFD methods is depicted in figure 1.10.[34]
There is no such thing as the perfect method that outperforms all the others in all scenarios but rather one method is more suitable than the other in a certain scenario. Generally speaking the conventional methods are more suited for macroscopic flow while the particle-based methods are mainly used for microscopic flow but mostly struggle with dense fluids. Additionally the latter are often tailored to a particular problem and used for research rather than being all-purpose built.

[33]Meshes that reflect the real geometry well enough and have good numerical properties, such as low asymmetry (skewness), and a corresponding numbering that leads to a linear equation system of diagonally dominant form.
[34]The methods that are not further discussed in this thesis are greyed out.

2. The Lattice-Boltzmann Method

Opposite to the traditional CFD methods, the aforementioned particle-based methods are not directly based on the continuum derived laws of conservation. One of the most popular due to its versatility are the Lattice Boltzmann methods (LBM), which have their roots in the *kinetic gas theory* and can be seen as the discretisation of the *Boltzmann equation*. The following chapter first presents in short the roots of the method in kinetic theory before explaining the basic algorithm for incompressible flow, its accuracy and stability as well as the implemented boundary conditions.

2.1. The theory behind Lattice-Boltzmann

The following section will give a short basic introduction into the kinetic theory of gases, the Boltzmann equation as well as the predecessors of Lattice-Boltzmann, the lattice gas automata. For further information on the subject refer to [1] and [6].

2.1.1. Kinetic theory of gases

Rather than using the continuum approach one might try to describe the underlying microscopic level: The *kinetic theory of gases* describes a dilute gas as a *large number of microscopic particles* in constant motion interacting with each other leading to processes that are characterized by *randomness*. Generally these models ignore the internal structure of the molecules and model the involved particles as structureless spheres that interact with each other and their environment only in elastic collisions.[35] Yet such simple models allow quantitative statements about the interactions in real gases: Macroscopic properties on a continuum level like pressure and temperature are derived statistically from the motion of those particles as well as transport laws and analytical solutions to the corresponding transport coefficients. The pressure for example is equal to the force exerted by the moving particles hitting an obstacle (see appendix C.2.1).

2.1.1.1. Knudsen number

The arguably most important number in the kinetic theory is the Knudsen number. The continuum description fails in micro-scale environments where the *mean free path of a molecule* between collision impacts λ_m is comparable to the *length scale* of the

[35]More sophisticated models also use far field interactions for a more realistic behaviour and non-elastic collision for modelling chemical reactions.

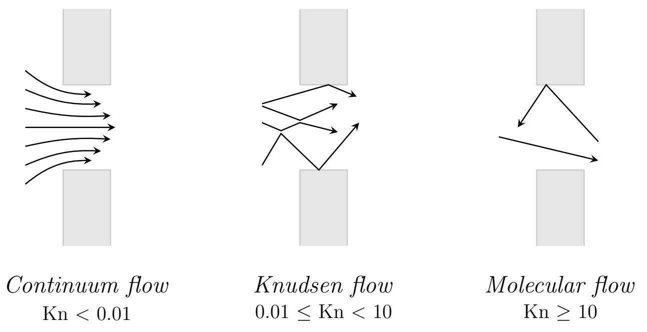


Figure 2.1.: Schematic flow for different Knudsen numbers

problem L[36] and statistical methods on microscale (e.g. kinetic theory of gases) must be employed. Hence the Knudsen number Kn is introduced as

$$Kn = \frac{\lambda_m}{L}.$$

Generally Knudsen numbers lower than $Kn \leq 0.01$ are considered continuum based flows, flows between $0.01 \leq Kn \leq 10$ are referred as transitional (rarefied gases) and for $Kn \geq 10$ collisions between particles can be neglected (molecular flow, figure 2.1). In the low transitional slip-flow regime (approx. $Kn \leq 0.1$) continuum mechanics methods may still be used with adequate boundary conditions that allow a certain slip at the wall while for larger Knudsen numbers the continuum derived methods fail. At all Knudsen numbers the walls are surrounded by a thin evaporation layer, the so called *Knudsen layer*, that might have to be modelled. [7]
The Knudsen number is closely connected to the Mach and the Reynolds number, given by the so called Kármán relation [8] and inversely proportional to density and characteristic length scale of the problem [1]

$$Kn \propto \tfrac{Ma}{Re}, \qquad Kn \propto \tfrac{1}{L\rho}.$$

2.1.2. Lattice gas automata

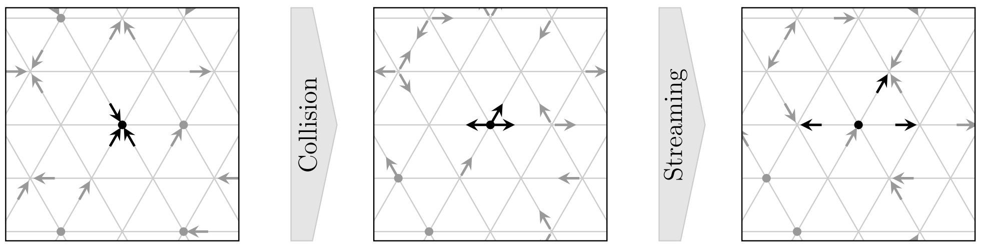


Figure 2.2.: The LGA algorithm: Collision and streaming of particles on a sparsely populated grid

[36]This can also be the case for super-sonic shock waves at $Ma > 1.5$

Already simple analytical kinetic gas models lead to accurate predictions in a real dilute gas. Analogously very simple numerical models can be used for simulating gas flow. One of them, the *Lattice gas automata* (LGA), a type of cellular automata,[37] can be seen as the direct *predecessors of LBM*. They were used to simulate rarefied gas flow back in the 1980s due to their simplicity and numerical efficiency.[38]

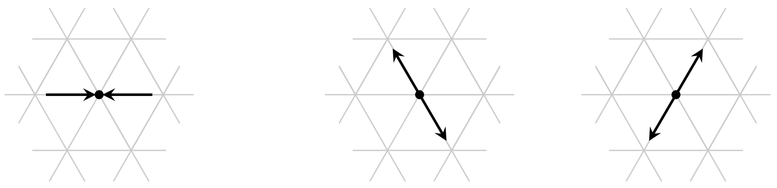


Figure 2.3.: An example for one of the non-deterministic collision rules in LGA: The collision on the left randomly leads to one of the distributions on the right.

The models consist of a sparse set of particles moving on a discrete grid, the so called lattice (figure 2.2). The particles are all *streamed to the neighbouring cells* in a time step according to their orientation on the grid.[39] According to some *non-deterministic collision rules* (the same input situation might yield multiple possible outcome states but only one is picked randomly, figure 2.3) [9] that ensure mass and momentum conservation the particles *collide* in each node, redistribute and the cycle starts again. The macroscopic quantities are determined using the particles' moments: The density is the sum of all particles at each node and the momentum is equal to the sum of the particles multiplied by the unit velocity. However those quantities are subject to a lot of *noise* due to the non-deterministic collision rules and have to be averaged over a large region to obtain reasonable results.

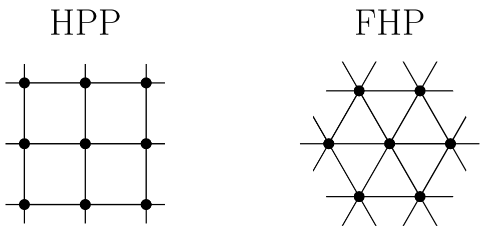


Figure 2.4.: The predecessor of LBM, the lattice gas automata: HPP (left) and FHP (right), both named after the acronyms of their authors

The first 2D model with a square lattice, the so called HPP model [10] lacked rotational invariance and was later replaced by the hexagonal grid FHP model. [11], [12] This improved version (figure 2.4) was less prone to Galilean invariance and anisotropy and could even simulate fluid flow in the continuum limit although restricted to very low

[37]A discrete model consisting of a grid of cells that can take discrete values and update values according to their neighbouring cells

[38]A single binary digit (boolean 0 or 1) per site and direction is sufficient to store the current state, in the case of the common FHP model this amounts to 6 bits per node.

[39]Particles only exist on the nodes of the grid and can inter-penetrate without collision while moving on the sides of the lattices.

Reynolds numbers. Nonetheless there have been issues porting the model to 3D: The square cube lacked invariance just like its two-dimensional counterpart and regular polytopes with a sufficiently large symmetry would lead to an dramatic increase in dimensions rendering the model inefficient[40] and overall limited.
In LBM therefore the effort was made to generalize the above algorithm by eliminating the non-deterministic rules using a statistical approach. The detailed chronological development from LGA to LBM can be found in [6].

2.1.3. Statistical mechanics

Describing all particles present in a dilute gas - let alone in a dense fluid - is not viable as the density of air under standard conditions is about 2.69×10^{19} *molecules*/*cm*^{3}.[41] Our scale of interest is in general though a lot bigger: Concepts like density, pressure and temperature don't even exist on the single particle level, they are the result of a huge number of particles interacting with each other. Thus one might try to describe an *ensemble of particles* using a stochastic approach. The basis of the Lattice Boltzmann methods and therefore the advantage over the Lattice gas automata is the elimination of the non-deterministic rules using *statistical mechanics*, the use of *probability theory in theoretical physics* to derive the average behaviour of a mechanical system. In the classical mechanics only a single state is considered while statistical mechanics introduces a statistical ensemble of states using *probability distributions* over the *phase space* $(\vec{x}, \vec{\xi})$.[42]

2.1.3.1. Particle distributions

The basis of kinetic theory is the introduction of a particle distribution function $f(\vec{x}, \vec{\xi}, t) = \frac{dN}{d\vec{x}\,d\vec{\xi}}$, where N denotes the number of molecules, which can be seen as a generalisation of the density ρ, which is only a function of time and space $\rho(\vec{x}, t)$. Particle distributions can be seen as 'densities' in time[43] and space as well as in a three-dimensional velocity space: The value of f equals to the density of particles at position $\vec{x}$ at the time t with the absolute velocity $\vec{\xi} = (\xi_x, \xi_y, \xi_z)$. [5]
Now all relevant macroscopic variables can be determined as *moments* of the distribution function through integration over the velocity space.[44] The zeroth moment equals to the *density*

$$\rho(\vec{x}, t) = m_P \int f(\vec{x}, \vec{\xi}, t) d\vec{\xi},$$

the first moment to the *momentum density*

$$\rho(\vec{x}, t)\vec{u}(\vec{x}, t) = m_P \int \vec{\xi} f(\vec{x}, \vec{\xi}, t) d\vec{\xi},$$

[40]All collision rules had to be saved in *look-up tables* and every single node had to be checked for the correct collision rule for the current configuration.
[41]One mole of any ideal gas contains 6×10^{26} molecules (Avogadro number) while occupying 22.4 litres at standard conditions.
[42]And herein lays the advantage that Lattice Boltzmann method has over solving the macroscopic Navier-Stokes equation regarding flow on microscopic scale.
[43]Implicitly through space and velocity
[44]In general any average value of a function ψ can be obtained by $\langle \psi \rangle = \frac{m_P}{\rho} \int \psi f d\vec{\xi}$.

where $\vec{u}$ is the mean macroscopic velocity. The second moment delivers the total energy density

$$\rho(\vec{x},t)e(\vec{x},t) = \frac{m_P}{2} \int \vec{\xi}^2 f(\vec{x},\vec{\xi},t) d\vec{\xi}.$$

which can be split up into energy due to bulk motion of the fluid $\frac{1}{2}\rho\vec{u}^2$ and the internal energy due to thermal motion, the *internal energy density*

$$\rho(\vec{x},t)e_i(\vec{x},t) = \frac{m_P}{2} \int |\vec{\xi} - \vec{u}|^2 f(\vec{x},\vec{\xi},t) d\vec{\xi}$$

where the relative velocity $\vec{\xi} - \vec{u}$ is often termed $\vec{v}$.

2.1.4. Maxwell-Boltzmann distribution

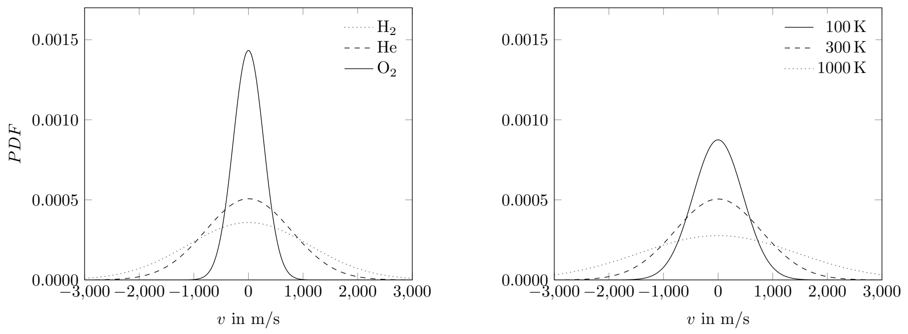


Figure 2.5.: One-dimensional Maxwell-Boltzmann distribution for different gases at room temperature (left) and helium at different temperatures (right) calculated using the formula below

It is only possible to construct such a distribution analytically for thermodynamic equilibrium (appendix C.3): Assuming symmetries in space and enforcing momentum conservation the so called *Maxwell-Boltzmann distribution function* can be found. The particle speed probability distribution indicates, which speeds are more and less likely in equilibrium depending on the temperature of the system and the particle mass (figure 2.5). First derived by Maxwell in 1860 and later elaborated extensively by Boltzmann in the 1870s, it is considered the basis of the kinetic theory of gases:

$$f^{(eq)}(\vec{x},|\vec{v}|,t) = n \frac{1}{(2\pi R_m T)^{\frac{D}{2}}} e^{-\frac{|\vec{v}|^2}{2R_m T}}.$$

D corresponds to the degrees of freedom which in the case of mono-atomic gases in three-dimensions are three. A simple derivation for this equilibrium distribution can be found in appendix C.3.2.

2.1.5. Boltzmann equation

For non-equilibrium, which is the general case, there is no analytical description available. But yet a transport equation that describes the *evolution* of a *probability distribution for position and momentum of a typical particle* can be formulated. Changes of different physical quantities transported by a rarefied fluid can be determined by this

so called Boltzmann equation, devised by the Austrian physicist Ludwig Boltzmann in 1872. This makes it more general than continuum mechanic based approaches, applicable to flow of all Knudsen numbers. The following derivation of the Boltzmann equation is based on [13].
In a real system an ensemble contains theoretically information about its state at any time but due to the enormous number of interactions over time this is converted into subtle correlations that appear chaotic and almost random. Boltzmann therefore assumed that the particles are un-correlated before collisions (one-sided molecular chaos). In dilute gasses it can be further assumed that the particles ($i = 1, \dots, N$) spend most of their lifespan on free trajectories apart from *collisions involving only two particles*[45] at a time for which in the case of mono-atomic gas[46] *classical Newtonian physics* can be assumed

$$\tfrac{dx_i}{dt} = \tfrac{p_i}{m}, \qquad \tfrac{dp_i}{dt} = F_i.$$

where p stands for the transferred momentum. In such a model the particle distribution would follow a simple transport equation, the Boltzmann transport equation. The total derivative of a distribution function f given by

$$\left.\frac{Df}{Dt}\right|_{transport} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x_i}\frac{\partial x_i}{\partial t} + \frac{\partial f}{\partial \xi_i}\frac{\partial \xi_i}{\partial t}.$$

must be equal to the changes caused by the collision.
This can be rewritten using $\partial x_i / \partial t = \xi_i$ and $\partial \xi_i / \partial t = F_i / \rho$ as well as introducing the notation $\Omega(f) = Df/Dt$ for the not yet defined *collision integral* to

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \nabla f + \frac{\vec{F}}{\rho} \cdot \nabla_{\vec{\xi}} f = \Omega(f)$$

This *non-linear integro-differential equation*[47] for the probability density function in six-dimensional space can't yet be solved unless the collision term is appropriately modelled. The exact collision operator proposed by Boltzmann (appendix C.3.4) is quite cumbersome but generally not all its properties have to be fulfilled: For hydrodynamic simulations only the first few momenta must be preserved during the collision. This allows for an enormous simplification of the collision operator as will be discussed later.

2.1.5.1. Derivation of the conservation equations and Chapman-Enskog expansion

The Boltzmann equation operates on a level of distribution functions. What happens on a *continuum level* with mass, momentum and energy is not directly accessible. Evaluating the first three moments of the conserved quantities it is possible to derive the *conservation laws* that *emerge* from the Boltzmann equation. This requires the following

[45] Collisions involving multiple particles are so rare they can be neglected.

[46] In contrast molecules consisting of several atoms can exhibit inner degrees of freedom and while the energy during collisions is always conserved it might be converted to rotational or vibrational energy leading to inelastic or super-elastic collisions. [5]

[47] 'Integro' as the original collision operator proposed by Boltzmann involves a collision integral (see appendix C.3.4).

integrals to be solved

$$m_P \int f d\vec{\xi} = \rho,$$

$$m_P \int \xi_i f d\vec{\xi} = \rho u_i,$$

$$\int \frac{\partial f}{\partial \xi_i} d\vec{\xi} = 0,$$

$$m_P \int \xi_j \frac{\partial f}{\partial \xi_i} d\vec{\xi} = -\int \frac{\partial \xi_j}{\partial \xi_i} f d\vec{\xi} = -\rho \delta_{ij},$$

$$m_P \int \xi_j \xi_j \frac{\partial f}{\partial \xi_i} d\vec{\xi} = -\int \frac{\partial (\xi_j \xi_j)}{\partial \xi_i} f d\vec{\xi} = -2\rho u_i,$$

$$m_P \int \xi_i \xi_j f d\vec{\xi} = \rho u_i u_j + \int v_i v_j f d\vec{\xi},$$

where $\vec{\xi} = \vec{u} + \vec{v}$.
Regardless of the precise form of the collision integral it has to be *orthogonal to any collision invariant*.[48]

$$\int \xi^k \Omega d\vec{\xi} = 0$$

Therefore we yield the corresponding continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0,$$

the momentum equation

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + g_i$$

as well as the energy equation

$$\frac{\partial (\rho e)}{\partial t} + \frac{\partial (\rho u_j e)}{\partial x_j} = \sigma_{ij} \frac{\partial u_j}{\partial x_i} - \frac{\partial q_j}{\partial x_j},$$

where the viscosity stress tensor σ_{ij} and the heat flux vector q_j are given by

$$\sigma_{ij} = -\int v_i v_j f d\vec{\xi},$$

$$q_j = -\frac{1}{2} \int v_i v_i v_j f d\vec{\xi}.$$

As we can see the stress tensor and the heat flux vector still depend on f and the corresponding integrals can't be evaluated if f is not known: The system is not closed.[49]
For thermodynamic equilibrium $f = f^{(eq)}$ the integrals containing f vanish and the equation system results in the Euler equations (appendix B.8). [7],[14]
The interesting part the Boltzmann equation is though describing non-equilibrium processes where f differs from $f^{(eq)}$. Close to the local equilibrium, at the continuum level,

[48]This reflects the conservation laws of mass, momentum and energy. A collision must not change the corresponding moment.

[49]We have no idea yet what the precise transport laws look like.

one might assume f as a *perturbed series* (appendix J) of the equilibrium distribution where the expansion parameter corresponds to the Knudsen number. This leads to a *hierarchy of approximations with an increasing deviation* of the distribution function *f from the thermodynamic equilibrium*. Recombining moments, taking into consideration their order of magnitude one can evaluate the corresponding integrals and is able to recover the full Navier-Stokes equation in the limit of small Knudsen numbers. A discrete version of this complex asymptotic method, called the *Chapman-Enskog expansion*, can be found in the appendix E.1.[15]

2.1.5.2. $\mathcal{H}$-Theorem

At the bottom of physics all laws are reversible according to Liouville's Theorem in classical mechanics and even the Boltzmann equation appears to be. This though seems to contradict our perception of the world. As a natural consequence Boltzmann tried to link the Boltzmann equation to a quantity that reflects the *irreversibility of a process*, the *entropy S* of a system by defining a quantity $\mathcal{H}$[50]

$$-S = \mathcal{H} := \iint f\, ln(f)\, d\vec{\xi} d\vec{x}.$$

This quantity $\mathcal{H}$ is not conserved, instead it decreases until the particle distribution f reaches equilibrium.[51] [17], [18]

$$\frac{d\mathcal{H}}{dt} \leq 0.$$

2.2. Incompressible LBM

As the Navier-Stokes equations emerge quite naturally from an asymptotic analysis of the continuous Boltzmann equation in the Chapman-Enskog expansion it seems obvious to *discretise* the Boltzmann equation and apply the same procedure. Historically this was mainly motivated by the aforementioned Lattice Gas Automata in addition to increasing computational power in the early 1990s.
Using low order discretisations and truncating the Maxwell-Boltzmann equilibrium distribution function for *low Mach numbers* it is possible to recover the *continuity* and *momentum* but not the energy equation with reasonable discretisations. Additionally, the momentum equation only leads to an error term instead of the correct dilatation. Therefore, this most common LBM algorithm is generally used for simulating *incompressible isothermal flow* [19] and the resulting error in the Mach number is kept small raising the viscosity in order to keep the Reynolds number constant. This effectively renders this particular Lattice Boltzmann algorithm a *weakly compressible fluid solver*: It solves the incompressible Navier-Stokes equations by allowing compressibility.[52]
Space and time are discretised just like in LGA but the method operates on the level of *representative clusters of particles* instead of individual particles, therefore, allowing much larger time steps than what an actual microscopic simulation would require. To reflect the almost random nature of the particles *stochastic* knowledge is included in

[50]This is equal to the common notation $S = k_B\, log(W)$.[16]
[51]This reflects the irreversable loss of information (appendix A.6)
[52]There do exist though a variety of similar numerical schemes that can even be applied to highly compressible flow (appendix F). So far though those schemes lack significant advantage over traditional CFD methods.

form of the collision operator rather than using non-deterministic collision rules. This kind of description that lies somewhere in between micro- and macroscale is called *mesoscopic* (figure 2.6). [20]

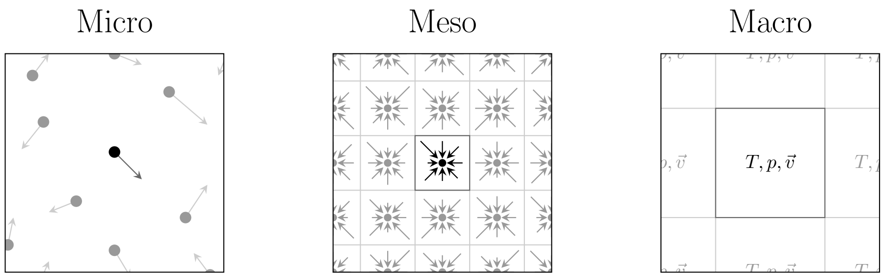


Figure 2.6.: The three different scales of fluid modeling: Micro-, meso- and macroscopic: Macroscopic models directly simulate collisions of particles while macroscopic models operate on basis of macroscopic parameters such as density, pressure, velocity and temperature.

Paradoxically this leads to a relatively simple algorithm which describes the advection of distribution functions and can be interpreted as *streaming and colliding of particle clusters across a finite number of nodes*. Due to its *simplicity* it is distinctly *faster than traditional CFD*, requires fewer resources and has huge advantages dealing with complex boundary conditions, implementing microscopic interactions and parallelisation.
The limiting factor is mainly the chosen *collision operator*. Due to its simplicity the Bhatnagar-Gross-Krook (BGK) collision model [21] is most commonly used but struggles in particular with high Reynolds number flow due to its highly viscosity-dependent stability. This led to a variety of more complex collision operators that relax hydrodynamic moments individually and *turbulence models* making this method even suitable for high Reynolds flow without triggering instabilities.
Regardless of the chosen fluid and despite the more complicated interactions between particles in a real fluid one is able to recover desired macroscopic equations (*top-down approach*) or approximate the microscopic level (*bottom-up approach*). This *duality* enables a broad usage of LBM in a lot of sectors. [20] As the method is still in development with thousands of papers published every year it is still not possible to set boundaries to its possibilities. Various successful applications have been reported over the years including turbulent flows [22]–[24], free-surface flows [25], non-Newtonian fluids [26], [27], multi-component [28] and multi-phase flows [29], fluid flow in porous media [30], micro-particles in fluid [31], blood flow [32], electrochemical systems [33], ion transport in nano-channels [34], reactive flow in fuel cells [35], fluid simulation in the non-hydrodynamic regime [36], [37] and many others.

2.2.1. Derivation of incompressible Lattice-Boltzmann

The incompressible Lattice-Boltzmann algorithm can be derived in multiple ways. Generally the asymptotic derivation is done using the Chapman-Enskog expansion as presented here or *Grad's Hermite expansion series* is applied [38]. In any case a particular collision operator has to be chosen for the derivation. The most common ones found in the literature are described below.

2.2.1.1. Collision operators

The biggest challenge in solving the Boltzmann equation consists in *simplifying the collision term*. The idea behind all common collision terms is *modelling the effect of the collision rather than the collision itself*: Every collision of a non-equilibrium system has the scope to relax it back to a local equilibrium which is assumed linearly. This approach is able to preserve the leading moments of the distribution that are needed to recover the macroscopic conservation equations.

Bhatanagar-Groos-Krook (BGK)
The best known model, the so called BGK-model, was introduced by Bhatnagar, Gross and Krook in 1954.[21] It is the classic LBM collision operator and due to its simplicity probably the main reason for the success of LBM. Up to now no other model is so *flexible* and can be used so *universally* but aöö this comes at the cost of reduced stability. The model is relaxed linearly towards a discretized Maxwell-Boltzmann distribution using a single relaxation time τ and is therefore also referred to as *Single-Relaxation-Time* (SRT) model. Instead of τ often its inverse, the collision frequency ω, is used.

$$\Omega = \frac{1}{\tau}(f_\alpha^{(eq)} - f_\alpha)$$

Normally the relaxation time is kept *constant* throughout the domain but more sophisticated versions introduce local tuning of the relaxation time. It should be emphasised that in any case even though the resulting equation might look linear it absolutely isn't as the local equilibrium term itself implicitly depends on the flow field.

Entropic collision operators (eLBM)
One problem with the BGK operator is that the *second law of thermodynamics* is not explicitly enforced: A system with BGK operator might lead to an increase in entropy locally and thus violate it. Entropic models, similarly to BGK, use a simple relaxation towards equilibrium but enforce the H-theorem rendering the scheme unconditionally stable but at a huge computational cost as generally an implicit equation for every node has to be solved. [39], [40], [41]

Multiple-Relaxation-Time (MRT)
The basic idea of Multiple-Relaxation-Time (MRT) models is transferring the collision to the *momentum space* using a transformation matrix and therefore *relaxing moments rather than populations*.[53] A general *collision matrix* with multiple relaxation times replaces the single relaxation time used in BGK and allows the moments' relaxation to be adjusted individually. Some of those are physically meaningful while others, called *ghost modes*, are not and can be used to enhance the model and get rid of numerical anomalies. Most commonly used is the model by D'Humières [42] where relaxation parameters of ghost modes are obtained from a linear stability analysis.[54]

[53]One can choose as many moments as discrete velocities. Generally the density and the momentum in every direction are chosen and additional moments for example for the stresses are introduced.

[54]The MRT collision operator degenerates to BGK if the diagonal values are all chosen identically and the off-diagonal ones are set to zero and reduces to TRT when the even and odd eigenvalues take specific values respectively. [43]

Two-Relaxation-Time (TRT)

Two-Relaxation-Time (TRT) models [44] can be seen as a special case of the MRT collision operators. *Even- and odd-order moments* in velocity are relaxed with two different relaxation rates ω^+ and ω^-. This can be achieved without a transformation to momentum space and therefore it combines the simplicity and efficiency of BGK with the accuracy and stability of MRT.
Populations can be *decomposed into symmetric and antisymmetric parts* according to

$$f_\alpha^+ = \frac{f_\alpha + f_{\bar{\alpha}}}{2}, \qquad f_\alpha^- = \frac{f_\alpha - f_{\bar{\alpha}}}{2},$$

$$f_\alpha^{(eq)+} = \frac{f_\alpha^{(eq)} + f_{\bar{\alpha}}^{(eq)}}{2}, \qquad f_\alpha^{(eq)-} = \frac{f_\alpha^{(eq)} - f_{\bar{\alpha}}^{(eq)}}{2},$$

where the overlined indices $\bar{\alpha}$ indicate the opposite direction of α. The rest population f_0 is attributed to f^+ while f_0^- and $f_0^{(eq)-}$ are zero respectively. The individual populations can be reconstructed by

$$f_\alpha = f_\alpha^+ + f_\alpha^-, \qquad f_{\bar{\alpha}} = f_\alpha^+ - f_\alpha^-,$$

$$f_\alpha^{(eq)} = f_\alpha^{(eq)+} + f_\alpha^{(eq)-}, \qquad f_{\bar{\alpha}}^{(eq)} = f_\alpha^{(eq)+} + f_\alpha^{(eq)-}.$$

resulting in discrete equation very similar to the BGK-Lattice-Boltzmann equation

$$f_\alpha(\vec{x} + \vec{c}_i \Delta t, t + \Delta t) = f_\alpha(\vec{x}, t) + \omega^+ (f_\alpha^{(eq)+} - f_\alpha^+) + \omega^- (f_\alpha^{(eq)-} - f_\alpha^-)$$

In this case only ω^+ is connected to the shear viscosity by

$$\nu = c_s^2 \Big(\frac{1}{\omega^+} - \frac{1}{2} \Big)$$

whereas ω^- is left as a free parameter.
Instead of adjusting ω^- generally the so called *magic parameter* Λ

$$\Lambda = \Big(\frac{1}{\omega^+} - \frac{1}{2} \Big) \Big(\frac{1}{\omega^-} - \frac{1}{2} \Big)$$

is altered as it has been linked to stability and accuracy. For the most stable simulation $\Lambda = \frac{1}{4}$ should be chosen whereas $\Lambda = \frac{3}{16}$ was shown to result in a wall location exactly in the middle between the walls and the fluid nodes for the case of Poiseuille flow. Keeping Λ fixed in between simulations furthermore eliminates the relaxation time dependent errors typical for BGK.

Central moments collision operators

This category of collision operators can be seen as an extension of the MRT collision operator: The collision step is also performed in the *momentum space* but in a reference frame *moving with the particle velocity* (referred to as central moments). In particular the cumulant collision operator proposed by Geier employs 9 discrete velocities in 2D and 27 in 3D in order to reduce the Galilean invariance exhibited by the traditional LBM models. [45], [46] Similar collision operators are reported to be used in the most common commercial LBM solver, Dassault System's XFlow. [47]

2.2.1.2. Chapman-Enskog expansion

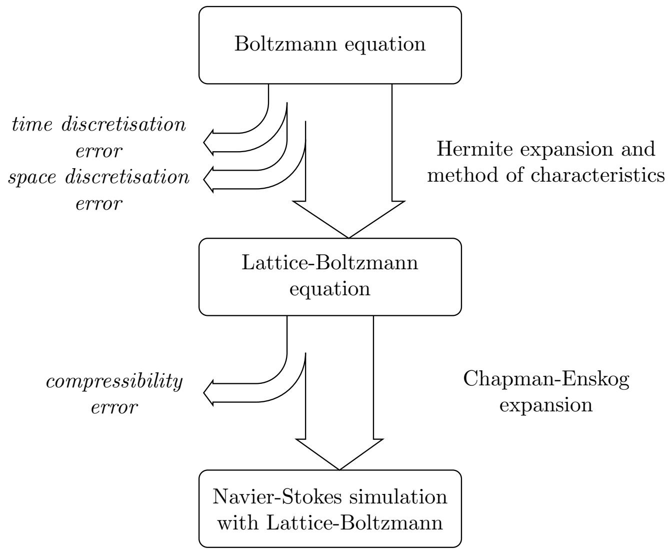

Boltzmann equation
time discretisation error
space discretisation error
Hermite expansion and method of characteristics
Lattice-Boltzmann equation
compressibility error
Chapman-Enskog expansion
Navier-Stokes simulation with Lattice-Boltzmann

Figure 2.7.: The steps in the incompressible Lattice-Boltzmann derivation

The derivation of the Lattice-Boltzmann method can be done in multiple ways which are all quite cumbersome. The traditional derivation applying the Chapman-Enskog analysis using a D2Q9 lattice is therefore only given in appendix E.1. The basic idea behind it consists in applying a *multi-scale expansion* in a small parameter ϵ which is generally seen as the Knudsen number as it appears in the non-dimensional Boltzmann equation (appendix C.3.5). Using a *perturbed series* to the equilibrium distribution function and combining conserved *momenta* one is able to yield equations resembling the continuity and momentum conservation with numerical artefacts of order $\mathcal{O}(Ma^2)$ but lacking the energy equation (figure 2.7).

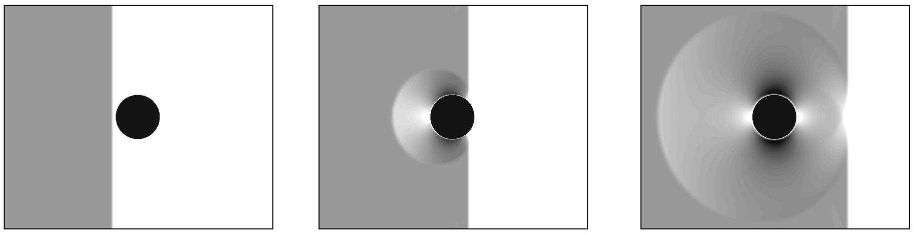


Figure 2.8.: Velocity magnitude around a solid sphere: A wave of constant velocity running into the obstable causes compressibility artefacts

This is no issue for the case of *isothermal incompressible flow* where the energy equation is decoupled. Therefore LBM is generally used for simulations in the limit of small Mach numbers where the additional error source apart from the obvious *discretisation errors* in space and time, the *compressibility error*, as well as the thermal decoupling does not matter. This renders LBM a quasi-compressible solver: It simulates incompressible flow by allowing compressibility but at the same time permits higher lattice velocities to be used and therefore a more coarse temporal resolution making the simulation particularly fast.[55] Therefore, LBM is plagued by *compressibility artefacts* like shock waves as can be seen in figure 2.8.

2.2.1.3. Energy equation

The problem with the basic $D2Q9$ lattice as well as with all other common lattices is that they lack particle speeds in order to be able to conserve the energy equation as well. One has to turn to more complicated multi-speed models [48], include local non-linear correction terms [49] or introduce *another set of distribution functions for advection-diffusion reflecting the energy equation* (chapter E.3) and possibly coupling the two lattices. This effectively doubles our number of distribution functions but is computational quite ineffective although relatively easy to implement. It should be noted that this requires a different Chapman-Enskog expansion for advection-diffusion and different boundary conditions, that can be found in [13].

2.2.2. Non-dimensionalisation

Generally in LBM special *lattice units* are chosen as simulation parameters: The side of a single lattice Δx as well as the simulation time step Δt are chosen to one for the sake of simplicity and the other parameters for the simulation are scaled accordingly. The *correlation* between the simulation and a particular *physical system* is made through *dimensionless, scale-independent numbers*. In case of the incompressible Navier-Stokes equations the only characteristic number of relevance is the *Reynolds number*: Other relevant numbers like the Mach and the Knudsen number are assumed small and their influence on the calculation can be neglected.
This leads to three layers of parameters : *Dimensioned* physical parameters must be converted into a *dimensionless* system by choosing appropriate characteristic measures (index d) and finally to a discrete simulation (lattice units, index lb) by choosing a discrete space and time step. [50], [51]
For the characteristic measures one may introduce the primary conversion factors for space c_L and velocity c_U according to

$$L_d = c_L\, L_{lb}, \qquad U_d = c_U\, U_{lb}, \qquad \rho_d = c_\rho\, \rho_{lb}.$$

Dividing the two equations the corresponding conversion factor for time c_T can be found to

$$T_d = \frac{c_L}{c_U}\, T_{lb} = c_T\, T_{lb}.$$

Through *dimension analysis* also other conversion factors like for the viscosity

$$\nu_d = c_U\, c_L\, \nu_{lb} = c_\nu\, \nu_{lb}$$

and the pressure

$$p_d = \frac{c_\rho\, c_L^2}{c_t^2}\, p_{lb} = c_\rho\, c_U^2\, p_{lb} = c_p\, p_{lb}$$

can be established.

[55] In my opinion one should think of LBM as a fictitious dilute model gas with good numerical properties that correlates with real system due to the laws of similarity.

2.2.3. The algorithm

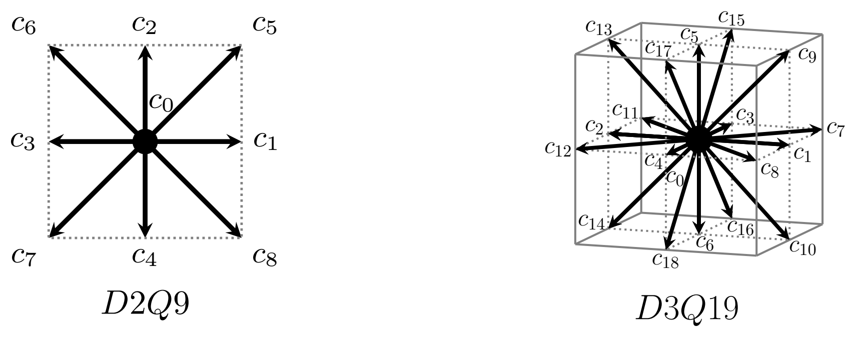


Figure 2.9.: The most commonly used LBM lattices: $D2Q9$ (left) and $D3Q19$ (right)

Using the Chapman-Enskog expansion one yields in case of the BGK operator a simple *evolution equation that preserves the Navier-Stokes equations on a macroscopic level*: The Lattice-Boltzmann equation[56]

$$f_\alpha(\vec{x} + \vec{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\vec{x}, t) + \frac{1}{\tau}(f_\alpha^{(eq)}(\vec{x}, t) - f_\alpha(\vec{x}, t))$$

where $f_\alpha^{(eq)}$ is the discrete Maxwell-Boltzmann distribution function given by

$$f_\alpha^{(eq)}(\vec{x}, t) = w_\alpha \rho \Big[1 + \frac{(\vec{e}_\alpha \cdot \vec{u})}{c_s^2} + \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{2c_s^4} - \frac{\vec{u}^2}{2c_s^2}\Big]$$

where the weights w_α depend on the chosen discretisation (common discretisations are presented in the next section).
The *equation of state*

$$p = c_s^2 \rho$$

links density and pressure for this model where the *lattice speed of sound*[57] is given for all common models by

$$c_s = \frac{c}{\sqrt{3}},$$

where

$$c = \frac{\Delta x}{\Delta t},$$

is the velocity emerging from the discretisation with Δx and Δt, which are usually chosen to 1 and the *viscosity* degenerates to a model constant that has to be tuned to match the physical Reynolds number

$$\nu_{lb} = \frac{(2\tau - 1)}{6} \frac{\Delta x^2}{\Delta t}.$$

Examples for common lattices are the $D2Q9$ and the $D3Q19$-lattice[58] (figure 2.9) that will be also subject to this thesis.

[56]This is one equation for every discrete direction α with a corresponding equilibrium distribution.

[57]Thermodynamically it has the significance of an isothermal speed of sound $c_s^2 = \left(\frac{dp}{d\rho}\right)_T = RT$.

[58]Both models involve a rest node for particles currently at rest.

This algorithm can further be split up into two steps:[59]
The *collision* step

$$f_\alpha^t(\vec{x}, t + \Delta t) = f_\alpha(\vec{x}, t) + \frac{1}{\tau}(f_\alpha^{(eq)} - f_\alpha)$$

in which the system is relaxed linearly towards a Maxwell equilibrium distribution and the *streaming step*

$$f_\alpha(\vec{x} + \vec{e}_\alpha \Delta t, t + \Delta t) = f_\alpha^t(\vec{x}, t + \Delta t)$$

where the distribution functions are streamed to the neighbouring cells (figure 2.10).

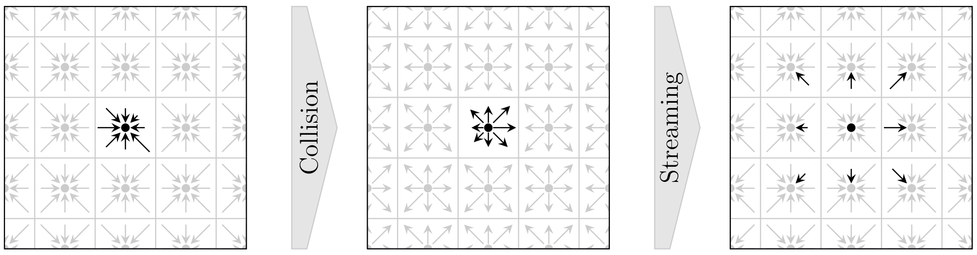


Figure 2.10.: The basic steps of the LBM algorithm: Collision and streaming

This lets us define the algorithm as follows (figure 2.11)[60]:

1. *Initialisation* (section 2.2.8) of the macroscopic values density ρ and velocity $\vec{u}$ and conversion to the mesoscopic distribution functions f_α (normally by assuming $f_\alpha = f_\alpha^{(eq)}$).
2. Calculate *macroscopic* density ρ and velocity $\vec{u}$ from the mesoscopic values of f_α using its moments
3. Compute the local *equilibrium distribution* function $f_\alpha^{(eq)}$ of a node
4. *Collision* step using the equilibrium distribution
5. *Streaming* step with the post-collision populations
6. Special treatment for *boundary conditions* (section 2.2.9)
7. Repeat steps 2 to 6

The *simulation time step can't be adjusted independently* but is rather set by the combination of characteristic velocity and characteristic length. This means short time steps are connected to smaller compressibility errors while larger time steps introduce larger compressibility errors. One should try to *keep the local Mach number*, defined by the local speed of sound, comparably *small* throughout the domain

$$Ma = \frac{U_{lb}}{c_s} \ll 1$$

[59]For an efficient implementation it is though common practice to use only one loop and do both steps at once

[60]Of course this algorithm can be modified to accommodate additional steps. In the case of no-slip boundaries using half-way bounce-back the algorithm can be changed to a single step that handles collision and streaming altogether and then the corresponding boundary cells are corrected by streaming backwards.

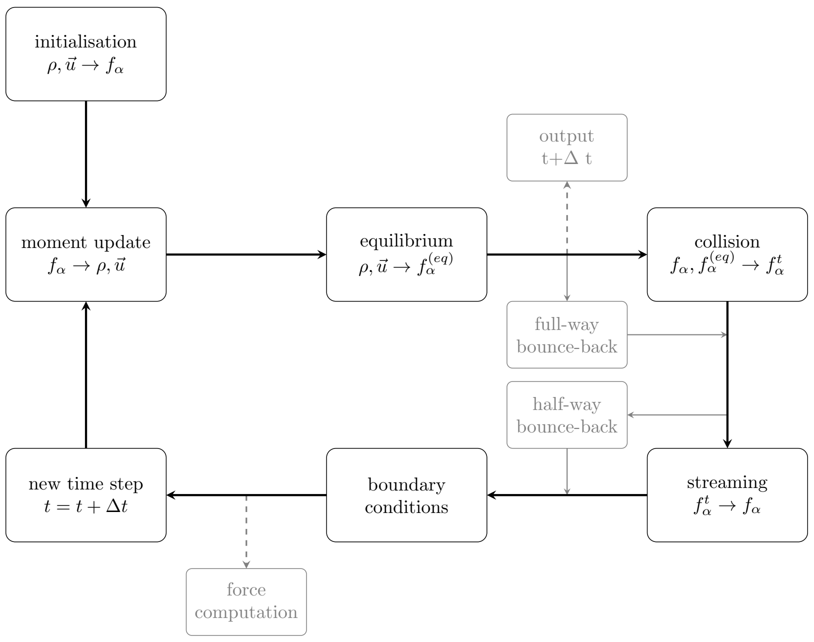

initialisation
$\rho, \vec{u} \rightarrow f_\alpha$
output
t+Δ t
moment update
$f_\alpha \rightarrow \rho, \vec{u}$
equilibrium
$\rho, \vec{u} \rightarrow f_\alpha^{(eq)}$
collision
$f_\alpha, f_\alpha^{(eq)} \rightarrow f_\alpha^t$
full-way
bounce-back
half-way
bounce-back
new time step
$t = t + \Delta t$
boundary
conditions
streaming
$f_\alpha^t \rightarrow f_\alpha$
force
computation

Figure 2.11.: Schematic LBM algorithm

with enough overhead for acceleration due to small channels such as in porous media.[61] The domain resolution is adjusted arbitrarily and enforcing Reynolds similitude with

$$Re = \frac{U_{lb} L_{lb}}{\nu_{lb}}$$

one yields the simulation viscosity ν_{lb}.

2.2.4. Discretisations

Generally the Boltzmann equation is discretised introducing what could be seen as a *explicit finite-difference discretisation on a regular cubic grid*. The corresponding discrete lattices are denoted in the *DdQq* notation where *d* corresponds to the number of dimensions in the problem and *q* is the number of symmetric finite velocities. Depending on the problem solved a *rest particle* with velocity 0 might be required leading to odd velocity sets.

2.2.4.1. D2Q9

The *D2Q9* model is arguably the most common 2D model for incompressible flow. The lattice and the corresponding possible orientations of boundary conditions can be

[61] I personally try to keep the maximal over-speed in the resulting domain $|\vec{u}|$ due to contractions at around 0.1.

seen in figure 2.12.
The *collision frequency* is given by

$$\omega = \frac{1}{3\nu + \frac{1}{2}}$$

and the *lattice velocities* can be calculated to

$$\begin{bmatrix} e_{\alpha x} \\ e_{\alpha y} \end{bmatrix} = c \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}$$

with the corresponding *weights*

$$w_\alpha = \begin{bmatrix} \frac{4}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} \end{bmatrix}.$$

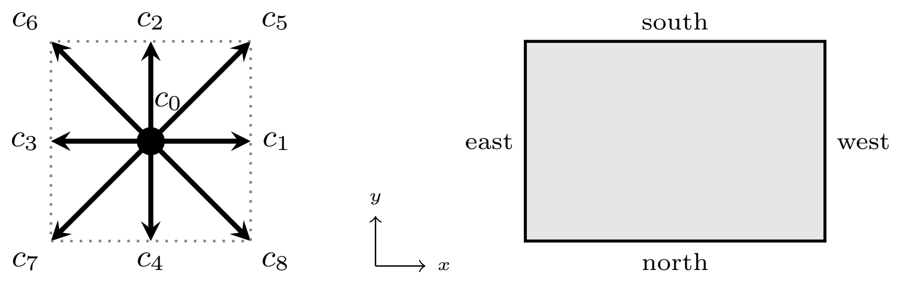


Figure 2.12.: The D2Q9 model and all possible boundary node locations for 2D

2.2.4.2. D3Q19

The *D3Q19* model is one of the most common models for simulation of 3D incompressible flow as it is about a third less computational heavy then the *D3Q27* model while delivering similar accuracy. The lattice as well as all possible boundary node locations are depicted in figure 2.13.

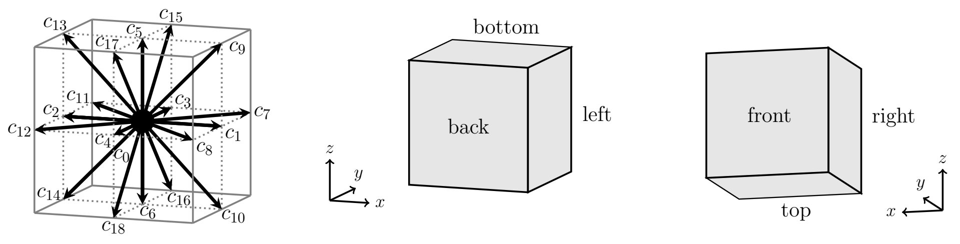


Figure 2.13.: The D3Q19 model and all possible boundary node locations

The *collision frequency* is given just like for the *D2Q9*-model by

$$\omega = \frac{1}{3\nu + \frac{1}{2}}$$

and the *lattice velocities* can be calculated to

$$\begin{bmatrix} e_{\alpha x} \\ e_{\alpha y} \\ e_{\alpha z} \end{bmatrix} = c \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

with the corresponding *weights*

$$w_\alpha = \begin{bmatrix} \frac{1}{3} & \frac{1}{18} & \frac{1}{18} & \frac{1}{18} & \frac{1}{18} & \frac{1}{18} & \frac{1}{18} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} & \frac{1}{36} \end{bmatrix}.$$

2.2.5. Macroscopic quantities

The scheme operates on the level of the mesoscopic distribution functions. The relevant macroscopic quantities can be determined as moments of those distribution function, similarly to kinetic theory (appendix C.3.3), according to table 2.1.

Moment	*Macroscopic property*
$\rho = \sum\limits_\alpha f_\alpha = \sum\limits_\alpha f_\alpha^{(eq)}$	Density
$\rho v_i = \sum\limits_\alpha e_{\alpha i} f_\alpha = \sum\limits_\alpha e_{\alpha i} f_\alpha^{(eq)}$	Velocity
$\tau_{ij} = \sum\limits_\alpha e_{\alpha i} e_{\alpha j} (f_\alpha^{(eq)} - f_\alpha)$	Viscous tress tensor
$p = \sum\limits_\alpha f_\alpha c_s^2 = \rho c_s^2$	Pressure

Table 2.1.: Relation between the mesoscopic moments and the macroscopic properties in Lattice-Boltzmann

2.2.6. Errors and accuracy

In traditional CFD methods it is generally possible to analyse the truncation error in the descretised equations directly and derive a formal order of accuracy. This is closely connected to the convergence of the numerical solution. In Lattice-Boltzmann though this cannot be done so easily as it is no direct discretisation of the Navier-Stokes equation.
Simulating incompressible flow with LBM one can identify *numerical* and *modelling errors* as the two main error sources (figure 2.14). The numerical errors emerge due to discretisation of the Boltzmann equation similarly to traditional methods and can be split up into three parts: [5]

- The *round-off error* emerges due to the fact that every computer is restricted to finite precision. It can be neglected if more significant digits (e.g. double precision with around 16 decimal digits) are used.[62]
- The *iterative error* is introduced due to the explicit time-marching.

[62]Some programs use large integer numbers instead as integer arithmetic on computers has certain computational advantages.

- The *discretisation error* emerges from the approximation of a continuous partial differential equation by a system of algebraic equations with a finite number of nodes.

As LBM is an artificial algorithm tuned in order to fulfil the incompressible Navier-Stokes equations another kind of error term emerges: the modelling error. It describes the deviation of our derived algorithm from the Navier-Stokes equations and originates in the low Mach number expansion of the equilibrium distribution function. This grid independent error limits LBM to the simulation of weakly compressible fluids and gets bigger as the Mach number increases. Thus, it is also referred to as *compressibility error*.

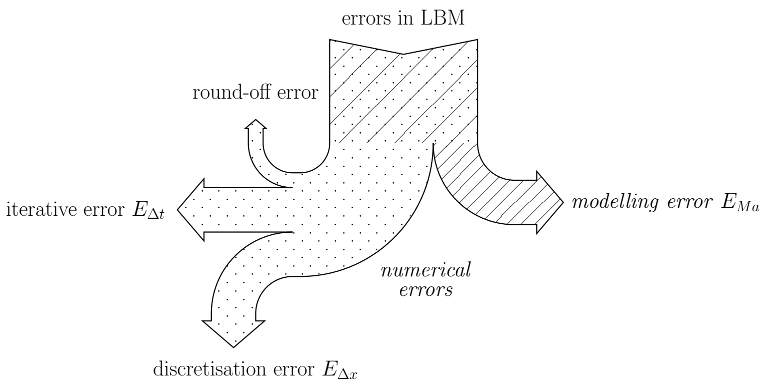


Figure 2.14.: The different types of error sources in LBM: The numerical errors are found in every discrete numerical simulation of a continuous system while the modelling error emerges from the low Mach number expansion.

As the round-off error can generally be neglected this leaves us with three main sources of error [50]: A *discretisation error* in space $E_{\Delta x}$ as well as in time $E_{\Delta t}$ and finally a *compressibility error* E_{Ma} linking the former two. While formally the governing equation looks like a first order explicit discretisation of the Boltzmann equation it can be shown through re-parametrisation to be actually equivalent to the second order discretisation,[5] meaning the discretisation errors scale with $\mathcal{O}(\Delta x^2)$ and $\mathcal{O}(\Delta t^2)$ respectively. This leaves us with an overall error according to

$$E = E_{\Delta x} + E_{\Delta t} + E_{Ma} = \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2) + \mathcal{O}\left(\frac{\Delta x^2}{\Delta t^2}\right).$$

This means the *space and time step can't be adjusted independently*: In order to achieve optimal results one should adjust them so the overall error is minimum. Refining the grid one would wish for a higher accuracy, so that the compressibility error won't take over. This implies that both errors, for space and compressibility, should be kept at the same order $E_{\Delta x} \sim E_{Ma}$ leading to:

$$\Delta t \sim \Delta x^2$$

The above stated relation reduces LBM though to an effectively only first-order accurate scheme in time. The *discretisation* even though being of *second order accuracy* can only *replicate the incompressible Navier-Stokes equations with a first order time accuracy*. [5], [50]

2.2.7. Stability

It might happen that a combination of certain input parameters and boundary conditions lead to a drift: *Exponentially growing errors of populations* and, therefore, also densities or velocities causing NaN ('Not a number') values of the relevant quantities. This is referred to as instabilities and in conventional CFD it is strictly linked to the *Courant number*. In LBM however the Courant number is always equal to unity and stability analysis is significantly more complicated: There is at least another degree of freedom available resulting from its kinetic nature; in case of the BGK operator the *relaxation time* τ. The inherently time-dependent Boltzmann equation can be seen more as an evolution equation that does not know, where it is heading.
Therefore, it is hard to predict the precise stability limits analytically but the major factor for instabilities stems from under-resolution: *Inadequately resolved grids* might cause divergent and unstable solutions especially for complex turbulent flow. In the case of single-relaxation models this can be attributed to *over-relaxation* of higher moments and can be partially avoided by turning to more complex collision operators. A main threat to the stability are *negative populations* that might occur with the basic collision operators: Theoretically only positive populations are valid but negative populations might appear with certain parameter combinations in nodes with high velocity gradients. Single negative populations may be cancelled out in the next collision step but multiple negative populations within a single cell can lead to a negative mass, and, therefore to a failure of the whole scheme. Equally too high velocities or pressure waves caused by incorrect *initial or boundary conditions* can cause an increasing compressibility error, leading the model to break down. Additionally most models' *insufficient Galilean invariance* is considered another important factor for divergent solutions. [52]
One distinguishes between *sufficient* and *necessary stability conditions*: Necessary stability conditions must be fulfilled in order to achieve stability but themselves do not guarantee a stable simulation: Other necessary stability conditions might have not been met. Sufficient stability conditions on the other hand guarantee a stable simulation. The least restrictive combination of necessary conditions is referred to as optimal conditions. [5]
One necessary stability condition in LBM is the *non-negativity of the viscosity* and hence for the BGK operator

$$\frac{\tau}{\Delta t} \geq \frac{1}{2}$$

has to be fulfilled. This is an important restriction: For the simple BGK operator stability and viscosity are linked. Choosing a more advanced model, like the two relaxation time (TRT), they can be decoupled. [53] Though the closer a particular relaxation time is to the stability limit the more likely is the simulation to crash due to emerging negative populations.[63]
For a system without any boundary conditions the *non-negativity of all equilibrium populations* is considered a sufficient stability condition and an optimal stability condition is found assuming non-negativity of the rest equilibrium population considering the range $\tau/\Delta t \geq 1$ leading to [54]

$$|\vec{u}| < \sqrt{\frac{2}{3}} \frac{\Delta x}{\Delta t}.$$

[63]For low velocities the lattice is rest node heavy: The rest node is the major population while the other populations are comparably small. Any small instability is dampened by the relatively large rest node population. For higher velocities this shifts and the algorithm becomes increasingly unstable.

2.2.8. Initial conditions

Generally the initial conditions are simply chosen as the *equilibrium distribution*. For unsteady, time-dependent problems in particular for time-dependent boundaries this might not be sufficient and it might be necessary to also initialise the non-equilibrium distributions correctly. This can be done using the *initialisation algorithm* according to Mei (figure 2.15). [55]

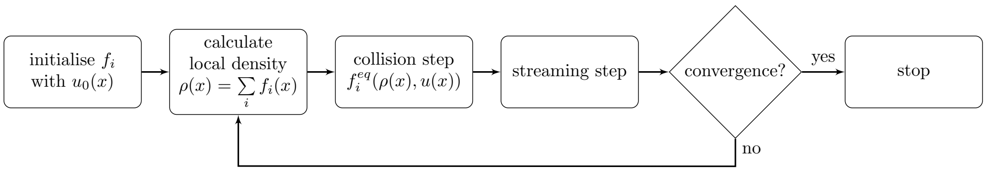

initialise f_i with $u_0(x)$
calculate local density $\rho(x) = \sum_i f_i(x)$
collision step $f_i^{eq}(\rho(x), u(x))$
streaming step
convergence?
yes
stop
no

Figure 2.15.: Schematic initialisation algorithm according to Mei

2.2.9. Boundary conditions

A crucial step for every numerical simulation is finding suiting boundary conditions. As *boundary conditions select solutions that are compatible with the external constraints*, a single inappropriate boundary may lead to errors and completely wrong results.

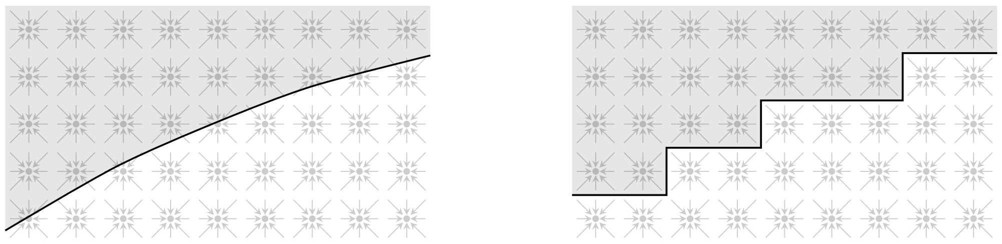


Figure 2.16.: A complex boundary (left) and the corresponding stair-cased simple boundary (right)

Unlike in conventional CFD methods the *boundary conditions* in Lattice-Boltzmann work rather counter-intuitively: Rather than working with macroscopic quantities like density and velocity one has to impose rules for the intangible mesoscopic *distribution functions*. Most of the time the arising equation system is *under-determined*: There is too little information given to reconstruct the distribution functions completely and hence certain *symmetries* for the unknown populations have to be assumed, meaning the boundary conditions have to be derived separately for every different lattice. Additionally boundary conditions in LBM are *not unique*: There exist a variety of methods for the same type of boundary condition. Due to the extremely fast algorithm one would *wish for simple and local boundary conditions* that do not significantly affect the computational speed.
Boundaries not aligning with the grid cutting mesh cells have to be taken into account either by simply *stair-casing* the boundary (figure 2.16) or using more sophisticated methods like extrapolation. [56] We will assume in the context of this thesis that every boundary can be stair-cased and modelled as a simple boundary. This is totally viable if the domain resolution is chosen fine enough to resolve the geometry adequately but

it has to be reconsidered whenever accuracy is crucial.

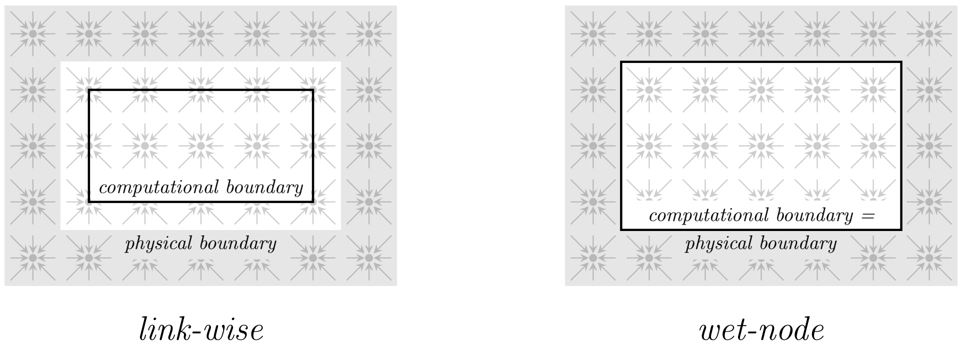


Figure 2.17.: The two main types of boundary conditions: link-wise and wet-node

Generally the boundary conditions are modelled as a special sort of collision between fluid particles and solid boundaries. There are two distinct approaches for boundary conditions. Either *simple reflection rules* for the populations are prescribed (*link-wise* methods) or populations are calculated from a *system of equations* (*wet-node* approach). Link-wise methods are algorithmically easy and naturally extend to different lattices but at the same time result in a shift of the actually boundary approximately half-way between solid and boundary nodes (figure 2.17). For wet-node boundaries the computational domain corresponds exactly to the physical domain but at the same time it is computationally more complicated and the set of equations must be derived for every single lattice individually. Additionally mass conservation is not exactly guaranteed in the case of wet-node techniques but is still consistent with the overall accuracy of the scheme. [5]

2.2.9.1. Transformation tables

The boundary conditions presented in the following section may be transformed to any lattice orientation with a simple transformation. This can be easily done using *transformation tables* that describe which index of the current orientation corresponds to which index in a given orientation. For the $D2Q9$ lattice the indices transform according to table 2.2 while the $D3Q19$ lattice transforms according to table 2.3.

location	*lattice velocities*	*velocities*
south	$[0,1,2,3,4,5,6,7,8]$	$[u_x, u_y]$
north	$[0,3,4,1,2,7,8,5,6]$	$[-u_x, -u_y]$
east	$[0,2,3,4,1,6,7,8,5]$	$[u_y, -u_x]$
west	$[0,4,1,2,3,8,5,6,7]$	$[-u_y, u_x]$

Table 2.2.: Transformation table for the $D2Q9$ lattice (reference configuration south)

location	*lattice velocities*	*velocities*
bottom	$[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]$	$[u_x, u_y, u_z]$
top	$[0,2,1,3,4,6,5,11,12,14,13,7,8,10,9,16,15,18,17]$	$[-u_x, u_y, -u_z]$
back	$[0,1,2,5,6,4,3,9,10,8,7,13,14,12,11,17,15,18,16]$	$[u_x, u_z, -u_y]$
front	$[0,1,2,6,5,3,4,10,9,7,8,14,13,11,12,16,18,15,17]$	$[u_x, -u_z, u_y]$
left	$[0,6,5,3,4,1,2,16,18,10,14,15,17,9,13,7,11,8,12]$	$[-u_z, u_y, u_x]$
right	$[0,5,6,3,4,2,1,15,17,13,9,16,18,14,10,11,7,12,8]$	$[u_z, u_y, -u_x]$

Table 2.3.: Transformation table for the $D3Q19$ lattice (reference orientation bottom)

2.2.9.2. Periodic boundaries

Periodic boundary conditions (figure 2.18) mean that distribution functions leaving the domain in one direction are re-entering from the opposite side. This is extremely useful in porous media where just a representative volume element is modelled and the rest of the domain is mirrored using periodic boundary conditions. Normally every Lattice-Boltzmann algorithm has implemented this method *generically*: The distribution functions are shifted circularly to the neighbouring cells and hence every cell at the border of a domain that isn't restricted by a boundary condition is automatically handled as a periodic boundary.

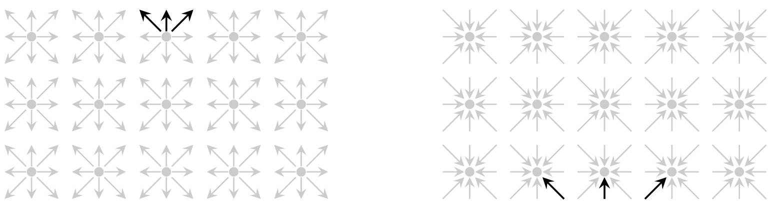


Figure 2.18.: Periodic boundary conditions: Populations before (left) and after streaming (right)

2.2.9.3. No-slip walls

In dense viscous fluids one can assume that the *fluid has zero velocity at any solid boundary*: The adhesion forces between the first layer of fluid particles and the wall will be larger than the cohesion forces between the particles.[64] This is referred to as no-slip wall and leads to the development of a boundary layer. In LBM no-slip walls are generally modelled by the so called *bounce-back* scheme due to its simplicity. It is simply assumed that incoming particles are reflected back into the domain. Algorithmically two possible ways can be realised: Either the streaming or the collision step are replaced by the bounce-back algorithm.

[64]The huge number of particles will lead to diffuse reflection.

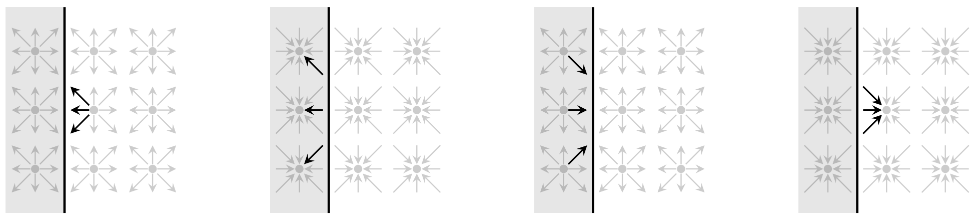


Figure 2.19.: No-slip wall modelled with full-way bounce-back method

In case of *full-way bounce-back* (on-grid bounce-back) populations travelling towards the wall leave the fluid domain and in the next time step the collision step for the first layer of boundary cells is replaced by a reflection of all populations (figure 2.19). Even though algorithmically easier this delay spanning two time steps degrades full-way bounce-back time accuracy to first-order. Alternatively in the case of *half-way bounce-back* populations travelling towards the wall are reflected in the same time step: The streaming step is replaced for the boundary cells by an inversion only of the relevant populations (figure 2.20).[65]

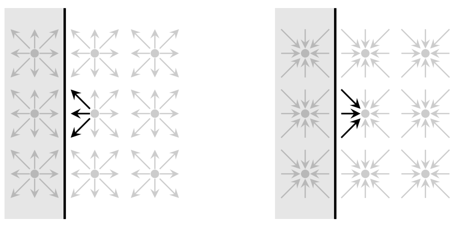


Figure 2.20.: No-slip wall modelled with half-way bounce-back method

Both approaches belong to the family of link-wise methods: By Chapman-Enskog analysis it can be proven that the boundary is located in between solid and boundary nodes. The exact position is varying depending on the viscosity if used with the single relaxation time BGK collision operator. [5] This can be interpreted as a *small tangential velocity* if the boundary is assumed half-way between the nodes which is a major drawback: In order for results to be comparable one has to ensure the same viscosity by varying the other parameters or by choosing another collision operator.
In case of *full-way bounce-back* the populations in the boundary change indices according to

$$\bar{\alpha}(\alpha) = [0,3,4,1,2,7,8,5,6]$$

for $D2Q9$ and

$$\bar{\alpha}(\alpha) = [0,2,1,4,3,6,5,12,11,14,13,8,7,10,9,18,17,16,15]$$

for $D3Q19$.

[65]I personally simply reverse the streaming step for the boundary cells after the combined collision-streaming step to achieve this behaviour.

2.2.9.4. Free-slip walls

For low Knudsen numbers as it is the case for very low pressure in rarefied gases the above mentioned principle of no-slip walls can't be applied. Instead it can be assumed that no momentum is exchanged with the wall along the tangential component: There is *no friction* between boundary and fluid, the tangential momentum remains unchanged and a simple elastic collision can be assumed (figure 2.21).

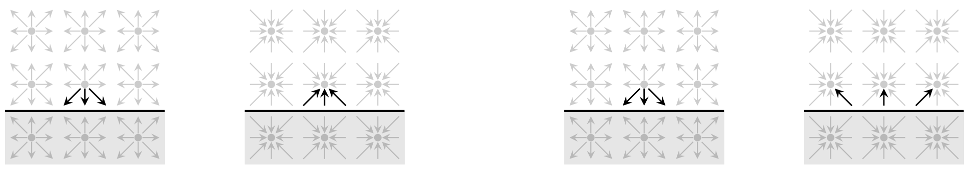


Figure 2.21.: No-slip (left) and free-slip boundary conditions (right)

2.2.9.5. Velocity and pressure boundaries

Both velocity and pressure boundary conditions can act either as in- or outlets depending on their orientation in the flow field. They pose no explicit restriction for the mass flow: Outflow can also happen on pressure inlet and backflow at pressure outlet boundaries.
As the LBM approach is only valid for incompressible fluids this links pressure to density rendering a pressure boundary condition basically a *density boundary condition*. Even though over the last couple of years other approaches have been suggested in literature for velocity and pressure boundaries [57] the most common ones for the $D2Q9$ and $D3Q15$-models are still the *Zou/He boundary* conditions introduced by Qisu *Zou* and Xiaoyi *He* in 1995 [58]. The emerging system of equations is under-determined and the *bounce-back of the non-equilibrium part of the distribution population* $f_i^{(neq)} = f_i - f_i^{(eq)}$ *normal to the boundary* has to be assumed. In 2008 Martin *Hecht* and Jens *Harting* expanded this approach to the *D3Q19* lattice [59]. As the resulting equation system is over-determined two additional variables, the so called transversal momentum corrections, N_x^z and N_y^z have to be introduced. Both of the above mentioned boundary conditions are reported to be *second order accurate* but if walls do not coincide with the lattice nodes the accuracy reduces to first order.
Here only the equations for the main orientations (south for 2D and bottom in the case of 3D) are given but, using the transformation tables presented in the previous section, boundaries with arbitrarily oriented boundaries can be generated.

D2Q9: Zou/He
For a boundary oriented at the bottom (orientation south), with f_4, f_7 and f_8 pointing into the wall, after streaming f_0, f_1, f_3, f_4, f_7 and f_8 are known and f_2, f_5, f_6 and the density ρ or the velocity u_y have to be determined. The equations available are the zeroth $\rho = \sum_\alpha f_\alpha$ and first momenta $\rho\vec{u} = \sum_\alpha \vec{e}_\alpha f_\alpha$ of the distribution functions:

$$\rho = \sum_{\alpha=0}^{8} f_\alpha = f_0 + f_1 + f_3 + f_4 + f_2 + f_5 + f_6 + f_7 + f_8,$$

$$\rho u_x = f_1 + f_5 + f_8 - (f_3 + f_6 + f_7),$$
$$\rho u_y = f_2 + f_5 + f_6 - (f_4 + f_7 + f_8)$$

where $u_x = 0$ in case of a pressure boundary.
As this system is under-determined an additional equation has to be introduced. This is done by assuming that the *bounce-back* rule still holds for the *non-equilibrium part of the particle distribution normal to the boundary*

$$f_2 - f_2^{(eq)} = f_4 - f_4^{(eq)}.$$

This yields in the case of a *velocity boundary*

$$\rho = \frac{1}{1-u_y}[f_0 + f_1 + f_3 + 2(f_4 + f_7 + f_8)]$$

and for a *pressure boundary*

$$u_y = 1 - \frac{1}{\rho_0}[f_0 + f_1 + f_3 + 2(f_4 + f_7 + f_8)]$$

with

$$f_2 = f_4 + \frac{2}{3}\rho u_y,$$
$$f_5 = f_7 - \frac{1}{2}(f_1 - f_3) + \frac{1}{2}\rho u_x + \frac{1}{6}\rho u_y,$$
$$f_6 = f_8 + \frac{1}{2}(f_1 - f_3) - \frac{1}{2}\rho u_x + \frac{1}{6}\rho u_y,$$

where $u_x = 0$ and $\rho = \rho_0$ in case of a pressure and $u_x = u_y = 0$ for a no-slip boundary.

D3Q19: Hecht/Harting

Martin Hecht and Jens Harting expanded this approach to 3D leading to the equation system:

$$\rho = \sum_{\alpha=0}^{18} f_\alpha,$$
$$\rho v_x = f_1 + f_7 + f_8 + f_9 + f_{10} - (f_2 + f_{11} + f_{12} + f_{13} + f_{14}),$$
$$\rho v_y = f_3 + f_7 + f_{11} + f_{15} + f_{16} - (f_4 + f_8 + f_{12} + f_{17} + f_{18}),$$
$$\rho v_z = f_5 + f_9 + f_{13} + f_{15} + f_{17} - (f_6 + f_{10} + f_{14} + f_{16} + f_{18}),$$

Now again bounce-back of the non-equilibrium part perpendicular to the boundary is assumed

$$f_5 - f_5^{(eq)} = f_6 - f_6^{(eq)}.$$

This equation system is now over-determined so another two variables, the *transversal momentum corrections* N_x^z and N_y^z, are introduced:

$$N_x^z = \frac{1}{2}[f_1 + f_7 + f_8 - (f_2 + f_{11} + f_{12})] - \frac{1}{3}\rho v_x$$

$$N_y^z = \frac{1}{2}[f_3 + f_7 + f_{11} - (f_4 + f_8 + f_{12})] - \frac{1}{3}\rho v_y$$

In case of a pressure boundary this yields

$$v_z = 1 - \frac{1}{\rho_0}[f_1 + f_2 + f_3 + f_4 + f_7 + f_8 + f_{11} + f_{12} + f_{19} + 2(f_6 + f_{10} + f_{14} + f_{16} + f_{18})]$$

and in the case of a velocity boundary this results in

$$\rho = \frac{1}{1 - v_z}[f_1 + f_2 + f_3 + f_4 + f_7 + f_8 + f_{11} + f_{12} + f_{19} + 2(f_6 + f_{10} + f_{14} + f_{16} + f_{18})].$$

And the unknown populations are then determined by

$$f_5 = f_6 + \frac{1}{3}\rho v_z,$$

$$f_9 = f_{14} + \frac{\rho}{6}(v_z + v_x) - N_x^z,$$

$$f_{13} = f_{10} + \frac{\rho}{6}(v_z - v_x) + N_x^z,$$

$$f_{15} = f_{18} + \frac{\rho}{6}(v_z + v_y) - N_y^z,$$

$$f_{17} = f_{16} + \frac{\rho}{6}(v_z - v_y) + N_y^z.$$

Again these equations can be transformed to arbitrary orientations using the transformation table presented in the previous section.

2.2.9.6. Pressure waves and non-reflecting boundaries

As Lattice Boltzmann is in fact a compressible fluid solver, sound waves might appear due to initial conditions as well as in unsteady and oscillating flow. Populations get reflected by walls and boundary conditions and spread wave-like until they are dampened out. *Every boundary* that enforces either constant density or constant pressure *reflects pressure waves*, and therefore affects accuracy negatively. In order to reduce the issue of reflected pressure waves there exist two different approaches: Several nodes thick *absorbing layers* absorb the pressure waves while *characteristic boundary conditions* are implemented around the proper boundary conditions and impose macroscopic values in such a way that no waves are reflected. [6], [13]

2.2.9.7. Corner treatment of boundary conditions

Particular care must be given to *corners and edges* of wet-node boundaries as they might degrade the overall quality of the simulation. In 2D (figure 2.22) for *concave corners* the problem is *under-determined* and additional constraints have to be introduced. *Convex corners* are *over-determined*, generally a simple bounce-back for the single unknown population is assumed. In 3D there are even more possible cases with edges and corners (figure 2.23). For the *D3Q19* lattice an approach is provided by Hecht and Harting. [59]

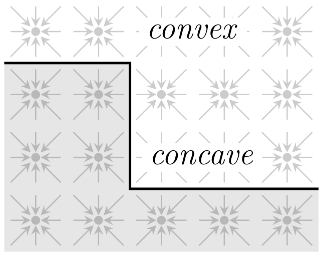


Figure 2.22.: Possibilities for corners in a $2D$ simulation

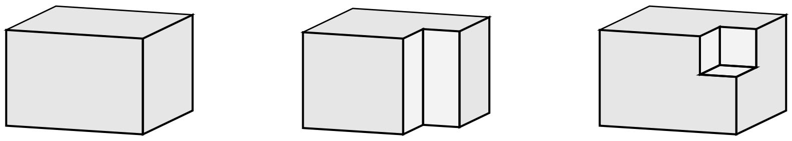


Figure 2.23.: $3D$ involves a lot more possible corner and edge orientations than its 2D counterpart

2.2.10. Forces on structures

Generally forces on structures are calculated using the *stress tensor* of the neighbouring cells. For half-way bounce-back a more straight forward approach might be used based on the transfer of momentum across the boundary, namely the momentum exchange algorithm.

2.2.10.1. Momentum exchange algorithm

The *momentum exchange algorithm*, short MEA, is based on calculating the change of momentum due to populations hitting the wall and bouncing back. Before every simulation the links between boundary and solid nodes have to be identified and at each time step the incoming $f_\alpha^{(in)}$ and outgoing populations $f_\alpha^{(out)}$ have to be determined (2.24).

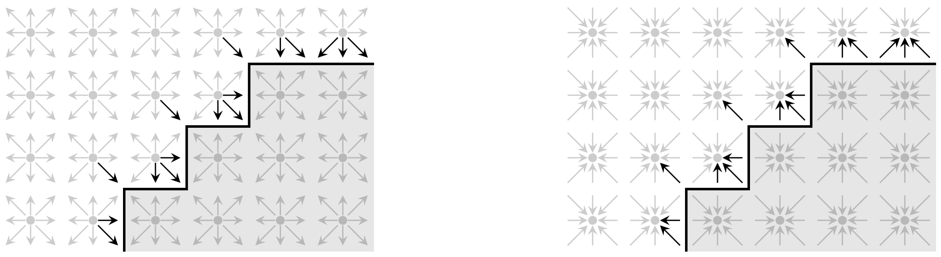


Figure 2.24.: Relevant incoming $f_\alpha^{(in)}$ (left) and outgoing populations $f_\alpha^{(out)}$ (right) for the momentum exchange algorithm

The change of momentum of a single direction can be calculated to

$$\Delta\vec{p}_\alpha = \vec{p}_\alpha^{(in)} - \vec{p}_{\bar{\alpha}}^{(out)} = f_\alpha^{(in)}\,\vec{e}_\alpha - f_{\bar{\alpha}}^{(out)}\,\vec{e}_{\bar{\alpha}}.$$

For a *stationary wall* modelled with half-way bounce-back obviously the incoming and outgoing populations are identical $f_{\bar{\alpha}}^{(out)} = f_{\alpha}^{(in)}$ and with $\vec{e}_{\bar{\alpha}} = -\vec{e}_{\alpha}$ the change of impulse simplifies to

$$\Delta \vec{p}_{\alpha} = 2 f_{\alpha}^{(in)} \, \vec{e}_{\alpha}.$$

The total change of momentum ΔP can be found by summing up all the contributions by every relevant direction α of every boundary link x_i^w:

$$\Delta \vec{P} = \Delta x^3 \sum_{x_i^w} (f_{\alpha}^{(in)} + f_{\bar{\alpha}}^{(out)}) \vec{e}_{\alpha}.$$

The corresponding force can then be calculated to

$$\vec{F} = \frac{\Delta \vec{P}}{\Delta t}.$$

This approach can be easily implemented into the standard LBM algorithm without slowing it down significantly delivering overall second order accurate results.

2.2.11. Turbulence models

Turbulence models in Lattice-Boltzmann play two important roles: They allow simulations of higher Reynolds number flow and they increase the viscosity and thereby the *stability* in particular by decreasing the effects of pressure waves.[66]

2.2.11.1. Smagorinsky turbulence model

Eddy viscosity models like the aforementioned Smagorinsky turbulence model can be implemented quite easily into LBM simulations.[67] Instead of the normal distribution functions a *filtered particle distribution* (just like the spatially filtered equations in LES) is introduced that is relaxed with a *locally tuned relaxation time*,[68] adjusted using the turbulent viscosity, to the local filtered equilibrium.
The overall viscosity

$$\nu = \nu_0 + \nu_T = \nu_0 + (C\Delta)^2 \, |S|$$

can be modelled introducing an additional local turbulent relaxation time

$$\tau = \tau_0 + \tau_T$$

since the viscosity can be decomposed as follows

$$\nu = \frac{(2\tau - 1)}{6} \frac{\Delta x^2}{\Delta t} = \frac{(2\tau_0 - 1)}{6} \frac{\Delta x^2}{\Delta t} + \frac{\tau_T}{3} \frac{\Delta x^2}{\Delta t}.$$

The filtered strain rate can be calculated to

$$|S| = \frac{|Q|}{2\rho c_s^2 \tau \Delta t}$$

[66]Theoretically this renders the numerical scheme limitlessly stable but pressure waves triggered by certain boundary conditions may still make a simulation crash.
[67]There are several different formulas that can be found online. Some are only valid for a density equal to unity and others define the Smagorinsky constant differently.
[68]In a similar way also non-Newtonian fluids can be modelled.

using the filtered momentum flux from the previous time step in good approximation

$$|Q| = \sqrt{\sum_i \sum_j \tau_{ij}\tau_{ij}}$$

where the non-equilibrium stress tensor τ_{ij} is given by

$$\tau_{ij} = \sum_\alpha e_{\alpha,i} e_{\alpha,j} \left(f_\alpha - f_\alpha^{(eq)} \right).$$

This results in the same algorithm as the standard incompressible LBM with an *additional local turbulent relaxation time* [60]–[62]

$$\tau_T = \frac{1}{2} \left(\sqrt{\tau_0^2 + \frac{2\sqrt{2}(C\Delta)^2}{\rho c_s^4 \Delta t} |Q|} - \tau_0 \right)$$

2.2.12. Advantages over conventional CFD

As seen in the previous chapter incompressible LBM boils down to a pretty simple algorithm that is *distinctly faster* than traditional incompressible CFD methods as it is *fully explicit* and *local*. The demanding *non-linear* collision step, can be performed *locally*, while the linear streaming step requires communication between neighbouring cells. This leads to excellent scalability and makes the algorithm ideal for massive *parallel architectures* such as graphic cards. On the other hand this renders LBM also very memory-intensive: Most of the time is spent copying values from the memory to the CPU's cache and a carefully chosen memory layout as well as proper data containers are required to make use of the computational efficiency.
Due to the equidistant grid unlike in traditional CFD methods *no meshing* process is needed, instead the domain is pixelised similarly to an image. This may lead to massive over-resolution in some parts of the domain but may be overcome using mesh refinement techniques (appendix F).
The kinetic background allows for simple local boundary conditions even in *complex geometries* and interactions on microscopic level, where kinetic effects can't be neglected. [63] The algorithm's roots in the kinetic theory are at the same time also its biggest drawback: LBM is *inherently time-dependent* and therefore not particularly well suited for steady flow simulations. This makes incompressible LBM perfect for transient simulations in porous media where this accurate spatial resolution is needed and meshing using traditional CFD may be difficult.

3. Multi-component flows

Most of them time rather than the sole hydrodynamic simulation *transport processes* are of interest. One important transport process is the diffusion of a species within another fluid. The following section explains the difference between multi-component and multi-phase flows first before going into the basics of these so called multi-component flows.

3.1. Multi-component and multi-phase flows

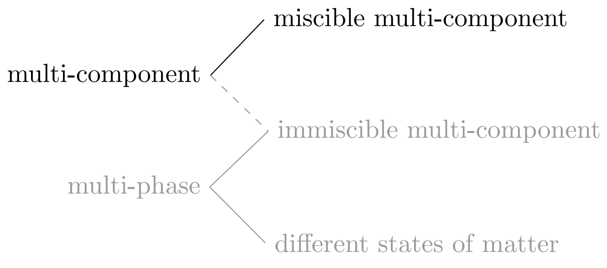


Figure 3.1.: Classification of multi-component and multi-phase flow

Flows of *different chemical species* that are *mixed on the molecular level* sharing the same velocity and pressure are referred to as *miscible multi-component* flow while flows involving immiscible components and/or different states of matter are generally termed multi-phase[69] flow. In the context of this thesis multi-component flows always refers to miscible multi-component flows while in the literature this might be used ambiguously (figure 3.1).
In multi-phase flow every phase has its own velocity field and momentum equation, and thus one needs to solve several momentum equations that interact with momentum exchange terms. In contrast in multi-component flow *the momentum of a single species is not conserved.* Instead the momentum between species is transferred in such a way that the total mixture momentum is preserved. [64]
In this thesis we will only focus on multi-component flow where only one fluid field is solved and the rate of transport of the species, given by their *mass fractions* (appendix D.2.2), is derived from the flow field using a *passive scalar*.

[69] A phase in this context refers to a region throughout which the physical properties of the material are uniform (appendix D).

3.2. Diffusion

One process common for systems involving inhomogeneity is *diffusion* where *substance moves from areas of high concentration to areas of low concentration* (appendix D.2.3)due to random molecular motion. Depending on the aggregate state, the number of species involved and the geometry this might involve a lot of different diffusion mechanisms and different models accounting for these phenomena might be deployed. Generally due to its simplicity some form of Fick's law is applied.[70]

3.2.1. Fick's laws

One way of describing diffusion are the empirically derived Fick's laws (appendix D.4). They assume that in steady state a flux establishes isotropically from sections with high concentration to areas with lower one, driven by the corresponding *concentration gradient* due to the random Brownian motion of particles. The diffusion mass flux $\vec{J}$ (appendix D.3) is assumed as proportional to the gradient of the *mass fraction of an individual component*

$$Y_i = \frac{m_i}{m}$$

according to[71]

$$\vec{J} = -\rho D_i \nabla Y_i.$$

Strictly Fick's laws are only valid for *dilute binary gaseous mixtures* that are not additionally driven by forces but they may be used for more complex flows with reasonable results. As for multi-component flows this approximation does not exactly conserve the mass, additional measures have to be taken. Commonly *all inconsistencies are absorbed into a major component* by solving the transport equations for all components but one and evaluating the mass fraction of this major component such that the conservation of mass is not violated[72]

$$\sum_i Y_i = 1.$$

3.2.1.1. Diffusion coefficients

The Fickian diffusion coefficients are a *material property* but as well a function of the concentration, *temperature and pressure*. Generally they increase with increasing temperatures and decrease with increasing density. Diffusion coefficients have generally to be found through experiments as there does not exist a single theory that is able to predict them accurately enough.
For *gasses* their magnitude is about $0.1 - 1 \frac{cm^2}{s}$, approximately inversely proportional to pressure and varies with the temperature to the power of $1.5 - 1.8$. Formulas derived from kinetic theory may predict the gaseous diffusion coefficients with an accuracy of about 8%. For *liquids* the values are significantly smaller with about $10^{-5} \frac{cm^2}{s}$ and may be predicted with an accuracy of about 20% by the Stokes-Einstein equation. In

[70]A short digression about the more general Maxwell-Stefan system can be found in appendix E.3.
[71]The derivation can be found in appendix D.4. Whenever this assumption fails to deliver accurate results, which might also be the case for porous media, the diffusion process is termed *non-Fickian*.
[72]More rigorous approaches include correction velocities to diminish this effect (appendix D.4.3).

the case of *solids* the diffusion coefficients are very small ($10^{-30}\frac{cm^2}{s}$) so that gaps and flaws like grain boundaries mainly account for diffusion effects. [65], [66]

3.3. Advection and diffusion

Most practical problems do not only involve diffusion but also *advection* either due to external agitation or caused by the diffusive motion.[73] In any scenario involving both mechanisms generally advection is assumed to be *superimposed* by diffusion: Diffusion is defined relative to the average velocity (figure 3.2).

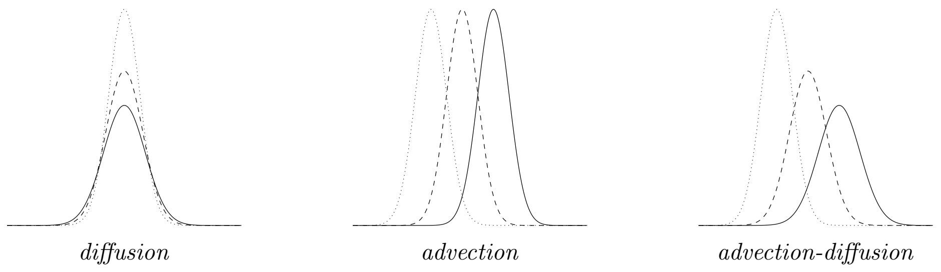


Figure 3.2.: Advection-diffusion can be seen as superimposed diffusion and advection

The *conservation equations for the mixture* with the corresponding mixture density, mass-averaged velocity, mixture-averaged kinematic viscosity and pressure remain *unchanged* (appendix D.2) while additionally the *mass conservation for each individual component*

$$\frac{\partial(\rho Y_i)}{\partial t} + \nabla \cdot (\rho Y_i \vec{u}) + \nabla \cdot (\rho Y_i \vec{u}_i) = \dot{\omega}_i.$$

must hold: The *mass fraction of an individual component* can be transported due to fluid flow with velocity $\vec{u}$, diffused with the *molecular diffusion velocity* $\vec{u}_i$ and changed due to chemical reactions with the production rate $\dot{\omega}_i$ within the system.
The sum of all species equations has to fulfil the mixture-averaged continuity equation. As atoms can't be generated due to the chemical reactions, this leads to the condition

$$\sum_i \nabla \cdot (\rho Y_i \vec{u}_i) = 0.$$

This equation system is *not closed*: The mass diffusion flux must be expressed as a function of the single species concentrations (single-fluid approach) or additional equations must be provided (multi-fluid approach). One possible way of closing the equation system are the aforementioned Fick's law.

3.3.1. Schmidt and Péclet number

With the Fick approximation and neglecting chemical reactions one might rewrite the advection-diffusion equation of a particular species to

$$\frac{\partial(\rho Y_i)}{\partial t} + \nabla \cdot (\rho Y_i \vec{u}) = \nabla \cdot (\rho D_i \nabla Y_i).$$

[73]In dilute solutions this self-induced convection may be neglected while for concentrated solution it has to be considered.

Assuming incompressible flow with constant material values throughout the flow field and introducing again the dimensionless variables one yields

$$\frac{\partial Y_i}{\partial t^*} + \vec{u}^* \cdot \nabla^* Y_i = \frac{D_i}{U\,L} \nabla^{*2} Y_i = \frac{1}{Re\,Sc} \nabla^{*2} Y_i = \frac{1}{Pe} \nabla^{*2} Y_i.$$

with the new dimensionless *Schmidt number*[74]

$$Sc := \frac{\nu}{D_i} \qquad \frac{\text{viscous diffusion}}{\text{molecular diffusion}}$$

which is related to another dimensionless number, the *Péclet number* for species transport by[75]

$$Pe := Re\,Sc = \frac{L\,U}{D_i} \qquad \frac{\text{advective transport rate}}{\text{diffusive transport rate}}.$$

As can be seen in particular from the exact solution of the Gaussian hill advection (section 6.1.4.1) *diffusion scales with square root of time* $\sqrt{t}$ whereas *advection is a linear process* $\propto u\,t$. Which of the two effects is predominant mainly depends on the spatial scale: Problems with a large Péclet number are advection dominated while for vanishing Péclet numbers diffusion prevails.

[74]For turbulent flow one must additionally consider a turbulent Schmidt number ($Sc_t = 0.7 - 1$) that accounts for turbulent diffusion due to the chaotic turbulent velocity fluctuations.

[75]The Péclet number is a dimensionless number relevant for any kind of transported quantities and may also be defined for other transport phenomena such as heat transfer as $Pe := Re\,Pr$. Additionally for thermal multi-component flow also another relevant characteristic number, the Lewis number Le, can be found (appendix D.5).

4. LBM for multi-component flows

For simulating miscible multi-component flows in LBM generally a second set of distribution functions preserving an *advection-diffusion equation* for the corresponding *mass-fractions* on a macroscopic level is introduced. This second lattice is purely passive: It does not have any effects on the Navier-Stokes equation and theoretically could be calculated after the simulation.[76]

4.1. Advection-diffusion model

Analogously to the energy equation (appendix E.3) one might use an advection-diffusion model to simulate species transport [67]

$$g_{\alpha,i}(\vec{x}+\vec{e}_\alpha \Delta t, t+\Delta t) = g_{\alpha,i}(\vec{x},t) + \frac{1}{\tau_i}\left(g_{\alpha,i}^{(eq)}(\vec{x},t) - g_{\alpha,i}(\vec{x},t)\right)$$

where each relaxation coefficient is linked to the corresponding diffusion coefficient by [68]

$$\tau_i = \frac{D_i^*}{c_s^2 \Delta t} + \frac{1}{2}.$$

The equilibrium distribution similarly to the incompressible algorithm is a truncated expansion of the Boltzmann equilibrium distribution which in this case can be truncated at first[77]

$$g_{\alpha,i}^{(eq)}(\vec{x},t) = Y_i w_\alpha \left(1 + \frac{(\vec{e}_\alpha \cdot \vec{u})}{c_s^2}\right).$$

or second order

$$g_{\alpha,i}^{(eq)}(\vec{x},t) = Y_i w_\alpha \left[1 + \frac{(\vec{e}_\alpha \cdot \vec{u})}{c_s^2} + \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{2c_s^4} - \frac{\vec{u}^2}{2c_s^2}\right]$$

and the *transported quantity* is the *mass fraction*

$$Y_i = \sum_\alpha g_{\alpha,i}$$

whereas the *velocity field is imposed externally* by the hydrodynamic lattice f_α.

[76]This basically is implied by the dilute species assumption.

[77]This is possible as fewer moments have to be preserved for recovering the corresponding macroscopic behaviour. Furthermore neglecting the linear velocity term leads to a simple diffusion model.

4.1.1. Accuracy, errors and correction terms

On a microscopic level this approach preserves the transport equation

$$\frac{\partial Y_i}{\partial t} + \nabla \cdot (Y_i \vec{u}) = D_i^* \nabla^2 Y_i + \frac{D_i^*}{c_s^2} \frac{\partial \nabla \cdot (\vec{u} Y_i)}{\partial t} + \frac{D_i^*}{c_s^2} \frac{\partial}{\partial x_j} \left(\frac{\partial (Y_i u_j u_k)}{\partial x_k} \right)$$

where the last two terms correspond to *error terms* and the last term is only present for a non-linear equilibrium distribution.

4.1.1.1. Linear equilibrium distribution

In case of a linear equilibrium distribution this error may be rewritten as

$$-D_i^* \frac{\vec{u}^2}{c_s^2} \nabla^2 Y_i.$$

The model suffers from an error that scales with $\mathcal{O}(Ma^2)$ reducing the scheme to first order. For a second-order approximation the inclusion of a forcing term was proposed that artificially cancels out the error term [67]

$$F_\alpha = \frac{w_\alpha}{c_s^2} \left(1 - \frac{1}{2\tau_i} \right) \vec{e}_\alpha \cdot \frac{\partial (\vec{u} Y_i)}{\partial t}$$

where the derivative can be estimated by *backward difference* using the values from the previous and current time step to

$$\frac{\partial (\vec{u} Y_i)}{\partial t} \approx \vec{u}(t) Y_i(t) - \vec{u}(t - \Delta t) Y_i(t - \Delta t).$$

4.1.1.2. Non-linear equilibrium distribution

In case of the non-linear distribution function the error term may be rewritten as

$$-\frac{D_i^*}{c_s^2} \frac{\partial}{\partial x_j} \left(\frac{Y_i}{\rho} \frac{\partial p}{\partial x_j} \right)$$

leading to a *velocity independent error*. A corresponding correction term is given by [67]

$$F_\alpha = \frac{w_\alpha}{c_s^2} \left(1 - \frac{1}{2\tau_i} \right) \frac{Y_i}{\rho} \vec{e}_\alpha \cdot \nabla p.$$

Chopard et al. [67] recommend the non-linear distribution function for fluid flow but the corresponding correction term above relies on a spatial extrapolation which is very unhandy for the final application in complex porous media.[78] Thus we will simply use the non-linear equilibrium distribution without any correction terms like recommended by Krüger [5] as the error unlike in the linear case is velocity independent.

[78]More complex models such as [68] introduce additional more complicated correction terms but increase the computational burden significantly. I think this diminishes the computational advantages of LBM drastically and therefore should be avoided.

4.1.2. Stability and TRT collision operator

In particular for very large and small Péclet numbers, similarly to very large Reynolds numbers, the BGK operator may suffer from instabilities. This might again be avoided by switching to the *TRT collision operator*. In this case the magic parameter should be chosen as $\Lambda = \frac{1}{6}$ for pure diffusion and as $\Lambda = \frac{1}{12}$ for advection. [5], [54] Unlike in the hydrodynamic case in the advection-diffusion model the *anti-symmetric collision frequency* ω^- is *linked to the diffusion coefficient* and the symmetric collision frequency ω^+ has to be calculated using the magic parameter Λ.[79]

4.1.3. Conversion to physical units

One correlates the Lattice-Boltzmann advection-diffusion to the physical system by additionally enforcing the identical *Schmidt number*. This is done by ensuring the same dimensionless group appearing in the advection-diffusion equation and thus by setting the LBM diffusion coefficient to

$$D_i = U\,L\,D_i^* = \frac{U\,L}{Pe} = \frac{U\,L}{Re\,Sc}.$$

This means one simply converts the dimensionless diffusion coefficient using the characteristic velocity and length of the LBM simulation to LBM units and adjusts the relaxation time accordingly.

4.1.4. Reduced lattices

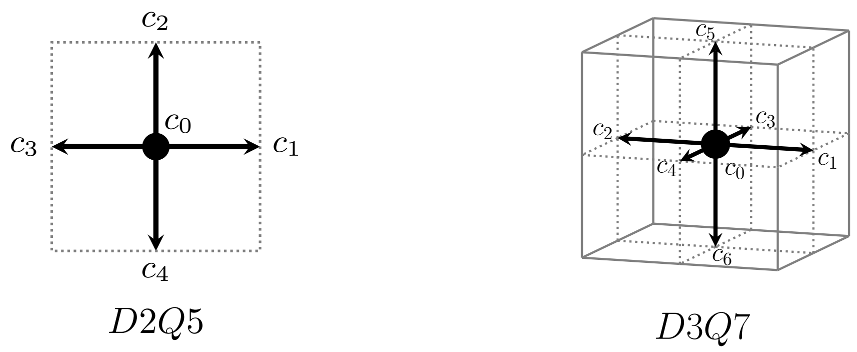


Figure 4.1.: The most commonly used reduced lattices: $D2Q5$ (left) and $D3Q7$ (right)

Commonly for advection-diffusion equations *lower order discretisations*, so called reduced lattices (figure 4.1), like the $D2Q5$ for $2D$ and the $D3Q7$ lattice for $3D$ are used with success as the fourth-order isotropic lattice tensors are not required for an adequate velocity set.[43], [69] They are able to successfully preserve the necessary moments even though communicating only with their nearest neighbours while cutting the *computational cost* in *half* and allowing for significantly *easier boundary conditions* as only one velocity points into each Cartesian spatial direction. In this thesis though they will not be used as the application will be complex porous media, where

[79]Thanks to Maximilian Gaedtke, MSc, admin at OpenLB, for pointing this out.

a consistent approach for diffusion and fluid flow is crucial.[80] For other applications in less complex geometries and for low Reynolds numbers it is recommended to use these lower order discretisations.

4.1.5. Boundary conditions

In the context of this thesis three different boundary conditions for the advection-diffusion equation are needed: An inlet, an outlet as well as the interaction with solid walls. As our final application will be porous media it is not viable to implement boundary conditions involving complex extrapolation approaches or boundaries that require the knowledge of the boundary normal. Therefore *only purely local boundary conditions* are discussed in this section.

4.1.5.1. Robin boundary condition

A Robin boundary condition is a boundary condition that imposes a *linear relation between a value of a function and its derivative*. This may also be the case at an inlet where the inlet flux of a particular component may be given by

$$\vec{J} \cdot \vec{n} = u_n Y_i - D_i \frac{\partial c_i}{\partial x_n}$$

where the index n denotes the direction normal to the boundary. A corresponding approach may be found in [70], [71].

4.1.5.2. Dirichlet boundary condition: Anti-bounce back

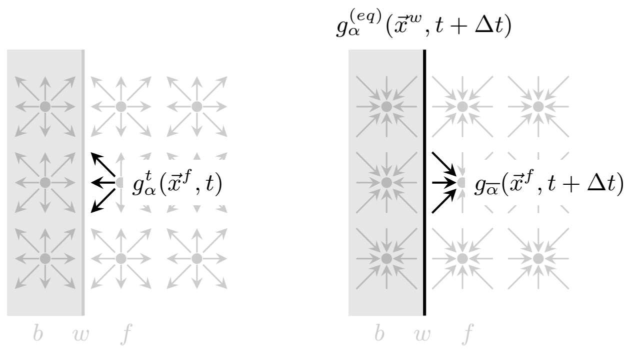

$g_\alpha^t(\vec{x}^f, t)$
$g_\alpha^{(eq)}(\vec{x}^w, t + \Delta t)$
$g_{\overline{\alpha}}(\vec{x}^f, t + \Delta t)$
b
w
f
b
w
f

Figure 4.2.: Schematic Dirichlet boundary condition based on anti-bounce-back

The inlet diffusion flux depends itself on the concentration gradient which can't be determined beforehand. For a fixed mass flow rate it must be disabled resulting in a condition that enforces only a certain *value of the solution* at this particular boundary, a so called Dirichlet boundary condition. In LBM a Dirichlet boundary condition is

[80]Imagine two $2D$ cells just barely touching each other with one corner and assume a fluid discretisation with a $D2Q9$ and an advection-diffusion discretisation with a $D2Q5$ lattice. The fluid discretisation would be able to interact across this gap while there could happen no diffusion due to the missing diagonal velocities.

generally implemented through anti-bounce-back algorithms.
The *anti-bounce-back* algorithm (figure 4.2) is almost as easy to implement as regular bounce-back boundaries. The reflected populations of neighbouring fluid cell for the next time step are calculated using the post-collision values that face towards the boundary and the corresponding symmetric equilibrium distribution in the boundary according to

$$g_{\bar{\alpha}}(\vec{x}^f, t + \Delta t) = -g^t_\alpha(\vec{x}^f, t) + 2g^{(eq)+}_\alpha(\vec{x}^w, t + \Delta t)$$

where for $\vec{x}^w$ corresponds to the values at the wall that can be determined by simple linear interpolation between the boundary and the neighbouring fluid node

$$Y^w = \frac{Y^b + Y^f}{2}, \qquad u^w_i = \frac{u^b_i + u^f_i}{2}.$$

For a stationary wall with the wall concentration Y^w_i the velocity terms in the equilibrium distribution vanish and the equation simplifies to

$$g_{\bar{\alpha}}(\vec{x}^f, t + \Delta t) = -g^t_\alpha(\vec{x}^f, t) + 2w_\alpha Y^w_i.$$

4.1.5.3. Neumann boundary condition

A Neumann boundary condition imposes a value for the *derivative* (flux) at a given point. In LBM this may be done by implementing a *Dirichlet boundary condition that embodies the Neumann boundary condition* by determining the cell value that the boundary cell must take in order to lead to the correct derivative through *extrapolation*.
The corresponding equation system is given by two Taylor series

$$Y_i(\vec{x}_0 + \Delta\vec{x}) = Y_i(\vec{x}_0) + \Delta|\vec{x}|\, Y'_i(\vec{x}_0) + \frac{\Delta|\vec{x}|^2}{2} Y''_i(\vec{x}_0) + \mathcal{O}(\Delta|\vec{x}|^3)$$

$$Y_i(\vec{x}_0 + 2\Delta\vec{x}) = Y_i(\vec{x}_0) + 2\Delta|\vec{x}|\, Y'_i(\vec{x}_0) + \frac{(2\Delta|\vec{x}|)^2}{2} Y''_i(\vec{x}_0) + \mathcal{O}(\Delta|\vec{x}|^3)$$

and the given derivative at the boundary cell

$$Y'_i(\vec{x}_0) = Y^{w\prime}_i.$$

From this equation system the value to be imposed by a Dirichlet boundary condition can be determined to

$$Y_i(\vec{x}_0) = \frac{4Y_i(\vec{x}_0 + \Delta\vec{x}) - Y_i(\vec{x}_0 + 2\Delta\vec{x}) - 2\Delta\vec{x} Y^{w\prime}_i}{3}.$$

For outflow often a *zero gradient* is assumed meaning the concentration remains constant through the outlet.[81] This leads to

$$Y_i(\vec{x}_0) = \frac{4Y_i(\vec{x}_0 + \Delta\vec{x}) - Y_i(\vec{x}_0 + 2\Delta\vec{x})}{3}.$$

A vanishing diffusion flux can also be mimicked by a concentration of the neighbouring fluid node equal to the boundary node $Y^f_i = Y^b_i$ and thus by evaluating the concentration before the outlet and imposing it on the outflow boundary. One might as well copy the post-collision populations to the outlet plane before propagation.

[81] For this condition to be valid the outlet has to be moved far enough away though.

4.1.5.4. Walls: Zero flux boundary condition

Walls impose a *zero flux condition in normal direction* but still allow a *diffusion flux in the tangential direction*: The tangential value may change along the boundary as a consequence of the concentration gradient. Even though criticised by some authors [72] this can be achieved by applying a simple bounce-back rule. [69] Krüger argues [5] this approach is correct as bounce-back rules only enforce a vanishing tangential velocity *in between* two nodes and thus this is no contradiction.

4.1.6. Advantages of LBM for advection-diffusion

For advection-diffusion problems most commonly finite difference and finite volume schemes are used. As pointed out by Wolf-Gladrow [73] LBM-based solvers additionally to higher computational speed can also achieve higher diffusion coefficients than explicit finite-difference schemes as LBM allows larger time steps than finite-difference solvers with the same spatial resolution. [5]

5. Porous media

Hydrodynamic flow and diffusion mechanisms in porous media are highly affected by size and structure of the pores. It is therefore useful to introduce different quantities for classifying porous media as explained in the following section.

5.1. Pore size, porosity and tortuosity

Porous media consist of *cavities* on micro or macro-scale *in a solid matrix*. One might classify them using their average pore width as well as their ratio between void volume to total volume.
Pore sizes below $2nm$ are generally considered *micro-pores* while the term *macro-pores* is used for pores greater than $50nm$. Everything in between is generally regarded as *meso-pores*. The ratio between void-space V_V and the total material volume V_T

$$\phi = \frac{V_V}{V_T}$$

is termed the *porosity* and is used in order to categorise the composition of a particular material. To describe how intricate the structure of a porous medium is, the *tortuosity* τ is introduced. In 2D it is generally defined as the ration between the length of a curve L to the distance between the ends

$$\tau = \frac{L}{D}.$$

In 3D there is no such clear definition available and generally reasonable estimates $\tau = 1.5 - 6$ or empirically derived formulas are taken. [65], [74]

5.2. Particle Reynolds number

For objects moving in a fluid or particle beds a particular kind of Reynolds number, the particle Reynolds number Re_p has to be used: For beds of spherical particles in a fluid the characteristic velocity is chosen as the unperturbed velocity in some distance of the sphere and the characteristic length as the diameter of the sphere. Under these conditions strictly *laminar flow* only exists for $Re_p \leq 10$. Non-spherical particles are generally either approximated by ellipsoids or the sieve diameter[82] is considered.
Additionally there exist other definitions of corrected particle Reynolds numbers that for example take into account the porosity of the medium. Other definitions might use the effective superficial velocity.[83]

[82]This is determined by a sieve analysis: The material is passed through sieves of progressively smaller mesh size and the average material stopped by a particular sieve is weighted.

[83]An average velocity defined as the ratio between volume flow rate and cross-sectional area.

5.3. Pressure drop in packed beds

For the horizontal flow through a bed of loose particles at low Reynolds numbers one might assume that the particles are not moved by the fluid flow: The gravitation force is bigger than the forces of the fluid flow. Due to the presence of media in the channel, blocking the domain and accelerating the flow, a *pressure drop between the inlet and the outlet*

$$\Delta p = p^{(in)} - p^{(out)}$$

is introduced. As for intricate geometries a simulation might not be feasible, a lot of calculations of porous beds still rely on approximate empirically derived formulas such as the Carman-Kozeny equation for laminar flow and the Ergun equation for turbulent flows presented in the following section.

5.3.1. Carman-Kozeny equation

Already Darcy observed in 1896 that the pressure drop for laminar flow of water through a sand bed of height h could be assumed proportional to the velocity

$$\frac{\Delta p}{h} \propto U.$$

For randomly packed beds of randomised spheres Carman and Kozeny found in 1937 the correlation for *laminar flow*

$$\frac{\Delta p}{h} = C_1 \frac{(1-\phi)^2}{\phi^3} \frac{\mu U}{L^2} \qquad Re_p \leq 1$$

where the characteristic length L corresponds to the diameter of a single sphere and the constant C_1 is generally chosen as 180.

5.3.2. Ergun equation

Ergun extended this equation in 1952 to *turbulent flow* by adding an additional term proportional to the square of the velocity

$$\frac{\Delta p}{h} = C_1 \frac{(1-\phi)^2}{\phi^3} \frac{\mu U}{L^2} + C_2 \frac{(1-\phi)}{\phi^3} \frac{\rho U^2}{L}, \qquad 3 \leq Re_p \leq 10^4.$$

In this case the constants are generally given by $C_1 = 150$ and $C_2 = 1.75$. Even though initially derived for spheres this can be extended to arbitrary particles by introducing a surface-volume mean diameter.[84]

[84]The diameter of a sphere with the same surface area to volume ratio.

5.4. Diffusion mechanisms in porous media

Similarly to continuum based flows also the traditional Fickian diffusion is based on the continuum assumption and may break down for dilute gas mixtures. This may also be the case in fine pore porous media. Depending on the mean free path of a molecule one distinguishes following diffusion mechanisms:

- *Molecular diffusion* happens for large macro-pores due to the thermal motion of the gas and can be described by a macroscopic Fick's laws.
- For meso-pores (such as for low pressure and small pore diameter) the mean free path is comparable to the scale of the system: A particle collides significantly more often with the wall than with another particle. In this so called *Knudsen diffusion* regime the diffusion coefficient is a function of the geometry.
- For micro-pores completely different mechanism come into play: For example an adatom[85] may hop around between adjacent adsorption sites on the surface depending on a variety of factors such as the bond between the adparticle and the surface (*surface diffusion*). A solvent might condense in pores and move around as a liquid and other solvents might dissolve in this liquid bubble and diffuse (*diffusion solubility*).[65]

In this thesis we will deal with macro-scale pores: Meso- and micro-scale phenomena will be neglected and only the molecular diffusion due to viscous flow will be considered.

5.4.1. Effective diffusion coefficient

Big parts of a porous medium are impermeable solid: The diffusion only occures in pores which themselves are not straight. In traditional CFD simulations the porous domain is often not resolved but instead just taken into account defining an *effective diffusion coefficient* such as

$$D_{\text{eff}} = \frac{\phi}{\tau} D$$

where ϕ and τ are the corresponding porosity and tortuosity of the sample. The diffusion takes place over a *longer distance* and a across a *smaller cross-section*.

5.5. Residence time

In particular for reactive flows in reactors the time matter spends inside, the so called *residence time*, and, thus the time for participating in reactions with other substances, is of uttermost importance. This section will elaborate which different quantities may be considered, how they can be determined experimentally and how they are connected to each other. For our scope the flow is assumed as incompressible and the only mechanism at the in- and outlet is advection and therefore the diffusive flux at these extremities of the domain is neglected.

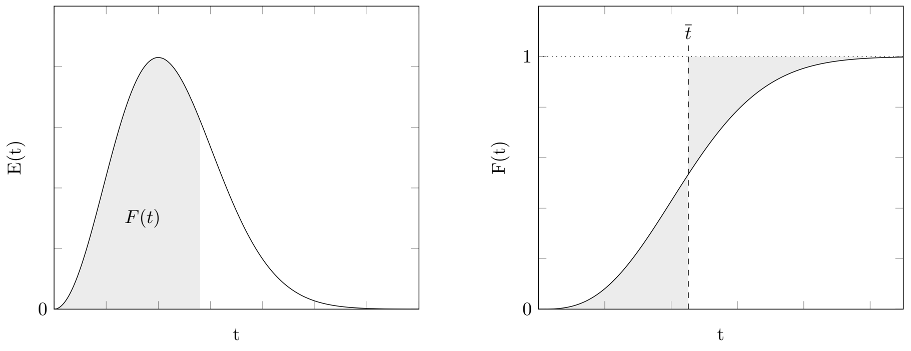


Figure 5.1.: Schematic residence time distribution (left) and its cumulative counterpart (right)

5.5.1. Residence time distribution

The main quantity defining the behaviour of chemical reactions in a reactor is the residence time, the time it takes for a fluid particle to move through the domain. A single particle has a single residence time while for more complex system this leads to a distribution: The real path of an arbitrary fluid particle might deviate drastically due to the formation of *dead waters* where the only transport possible is due to diffusion and *short circuits* where material passes through the domain significantly faster. Additionally backmixing might occur, where material flows backwards locally. For this reason a *residence time distribution* $E(t)$, short RTD (figure 5.1), is introduced using the flux of the amount of substance at the outlet and the amount of substance at the inlet according to

$$E(t) = \frac{\dot{n}_i^{(out)}(t)}{n_i^{(in)}}$$

where $E(t)dt$ characterises the probability of finding a particle that enters the domain at a time t and exits the domain within a time $t + dt$. Thus $E(t)$ is equivalent to the fraction of particles with a residence time between t and $t + dt$ and is sometimes referred to as exit age distribution. This leads to a very important property: All particles are expected to exit the domain for $t \to \infty$ and therefore

$$\int\limits_{t=0}^{\infty} E(t)\, dt = 1$$

has to be fulfilled.
The *average residence time* can be determined as the first moment of the residence time distribution

$$\bar{t} = \int\limits_{t=0}^{\infty} t\, E(t)\, dt$$

Another important mean time is the *turnover time* τ (hydrodynamic residence time for liquids) simply defined by the volume flow rate $\dot{V}^{(in)}$ and the reactor volume V

$$\tau = \frac{V}{\dot{V}^{(in)}}.$$

[85]A foreign atom trapped on a surface of a rigid body.

A mean residence time $\bar{t}$ significantly faster than the turnover time τ is a sign of stagnant fluid.

5.5.2. Cumulative residence time distribution

Experimentally another distribution, the *cumulative residence time distribution* $F(t)$, that is connected to the RTD by

$$F(t) = \int E(t)dt$$

is easier determined. Through numerical differentiation of the resulting distribution the RTD is recovered according to

$$E(t) = \frac{dF(t)}{dt}.$$

Therefore as a fundamental property this function must fulfil $F(t = 0) = 0$ and $\lim_{t\to\infty} F(t) = 1$.

5.5.3. Experimental measurement

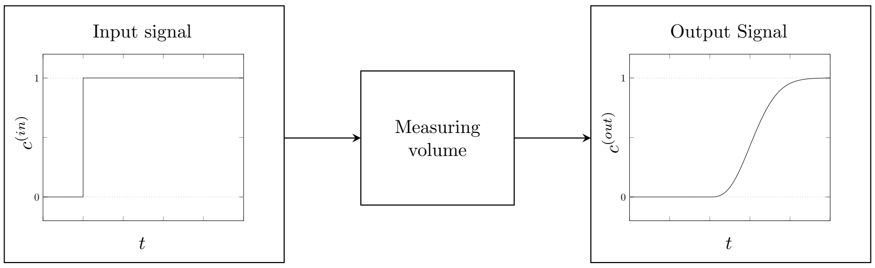


Figure 5.2.: Experimental measurement of the cumulative residence distribution: At the inlet the mass fraction is controlled to fulfil a Heaviside step function (left), the measuring volume distorts this signal, which is determined by measuring the outlet concentration over time (right).

For an ideal system all fluid elements leave the reactor in the same order they entered without mixing with particles in front of them and behind. The time it takes the fluid to leave the domain is characterised by a single residence time: If the system is fed with a certain concentration of a particular component at the inlet the same concentration will be measured at an arbitrary outlet after the turnover time. In a real system diffusion, a non-uniform velocity profile and turbulence might lead to a dispersion and a corresponding residence time distribution. In order to experimentally determine the residence time distribution $E(t)$ directly a Dirac delta function must be generated. For a limited time window, that must be significantly slower than the mean residence time, the domain must be flooded with a particular marker substance and the concentration at the outlet must be measured. This may be quite challenging and therefore instead the *cumulative RTD* is determined. For this purpose the tracer is let in rapidly but continuously (*Heaviside step function*) and the the *concentration at the outlet is tracked*. In such a case the cumulative RTD can be determined by

$$F(t) = \frac{n_i^{(out)}(t)}{n_i^{(in)}}.$$

5.6. Porous LBM: Sparse domain optimisation

For few boundary cells within a domain it does not pay off to create a logical mask and perform collision and streaming only for the relevant cells but instead it reduces performance.[86] But in a porous medium most nodes are not active parts of the algorithm and would therefore slow down the performance significantly. Therefore, the algorithm is modified so that cells that *do not contribute* to the simulation, such as solid cells not interacting with the fluid, are *excluded* from the simulation algorithm. In order to not keep track of the neighbours, the cells still take up storage and might be even pre-cached but are not going to be explicitly used in the algorithm. In this particular case[87] this boosts the algorithm to almost *double the speed* compared to the standard approach.

[86]As the populations in the wall still collide in the basic algorithm this might lead to very high velocity values though, in particular when using a turbulence model, which could possibly lead to confusion in post-processing.

[87]The porous domains in this thesis consists of up to 60% boundary cells of which only a fraction is needed.

6. Implementation

In the following sections two- and three-dimensional simulations obtained by LBM with the BGK and TRT collision operators as well as their corresponding counterparts with a Smagorinsky turbulence model are compared to various benchmark scenarios found in the literature for steady and transient flow. The model is then extended to binary multi-component flows using a second lattice for advection-diffusion with BGK and TRT collision operator. Furthermore the computational performance and the parallel scalability of the presented code is examined in detail and compared to the open source LBM solver Palabos. At the end of the chapter this verified model is used to simulate the binary multi-component flow through a porous bed and the results are compared to an Ansys Fluent simulation.

6.1. Single component flow

For single-component flow the standard benchmark scenarios of a lid-driven cavity flow as well as the flow around a cylinder at different Reynolds numbers in 2D as well as in 3D are considered as results are widely available in the literature and in publications. In the following section pressure and velocity boundary conditions are modelled with the Zou/He boundary conditions for 2D and the Hecht/Harting boundaries for 3D whereas the solid walls are modelled using half-way bounce-back.

6.1.1. Test case 1: Lid-driven cavity

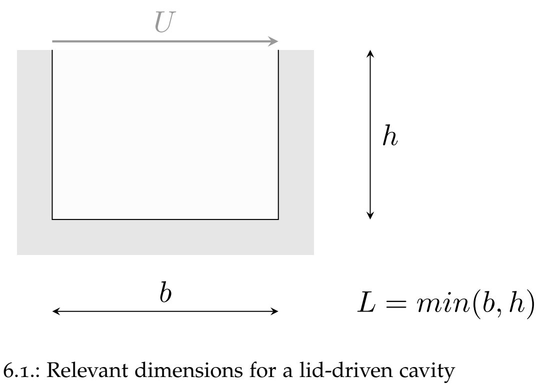


Figure 6.1.: Relevant dimensions for a lid-driven cavity

The lid-driven cavity is due to its simplicity a popular benchmark scenario for *steady fluid flow*. It consists of a cavity constrained by no-slip walls and a lid moving with a constant tangential velocity (figure 6.1). The characteristic length is generally given by the smaller dimension of the cavity `min(b,h)` and the ratio between $\frac{b}{h}$ is referred to as the aspect ratio. Due to the external force the liquid is pushed against the solid wall leading to different eddies depending on the Reynolds number and the aspect ratio. In the literature CFD-based values for velocity, pressure and vorticity can be found for a wide range of Reynolds numbers.

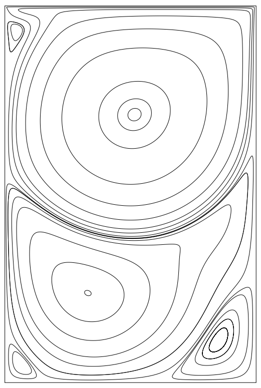


Figure 6.2.: Streamlines for a lid-driven cavity with an aspect ratio of 1.5 at Re 3200 calculated by a TRT-LBM simulation

The two-dimensional flow is simulated with a fluid initially at rest and compared to the corresponding literature for the case of a rectangular cavity [75] as well as a cavity with aspect ration 1.5 (figure 6.2) [76]. For all rectangular cavities a grid of 800 by 800 pixels was chosen and a comparably large characteristic velocity of $U = 0.1$. Only the results for the corresponding TRT collision operator are depicted in the following figures as the BGK collision operator would yield indistinguishable results. Overall the results are virtually identical with the data provided for all the Reynolds numbers considered (compare to figures 6.4 and 6.5) apart from Re 7500 where the solver remains stable but only a transient and no steady-state solution is found. The corresponding pressure and vorticity can be found in appendix H.1.

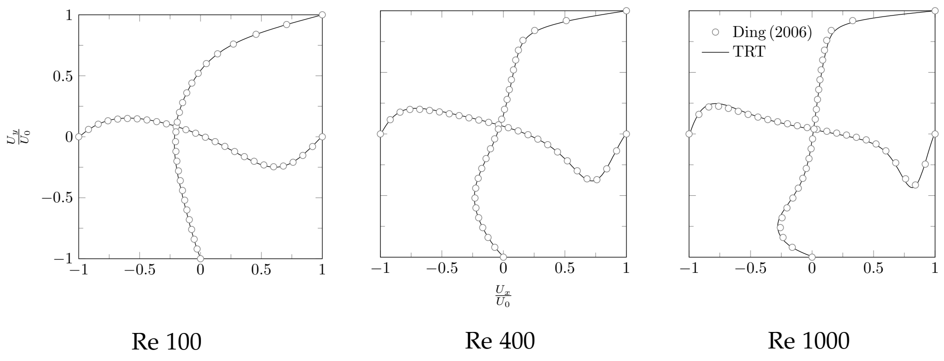


Figure 6.3.: Velocity cross-sections through the geometric center of three-dimensional cubic lid-driven cavities at different Reynolds numbers

The same is done for the case of a cubic three-dimensional lid-driven cavity where the velocity distribution deviates slightly from its two-dimensional counter-part. The results match those in [77] and are identical with [78] (figure 6.3).

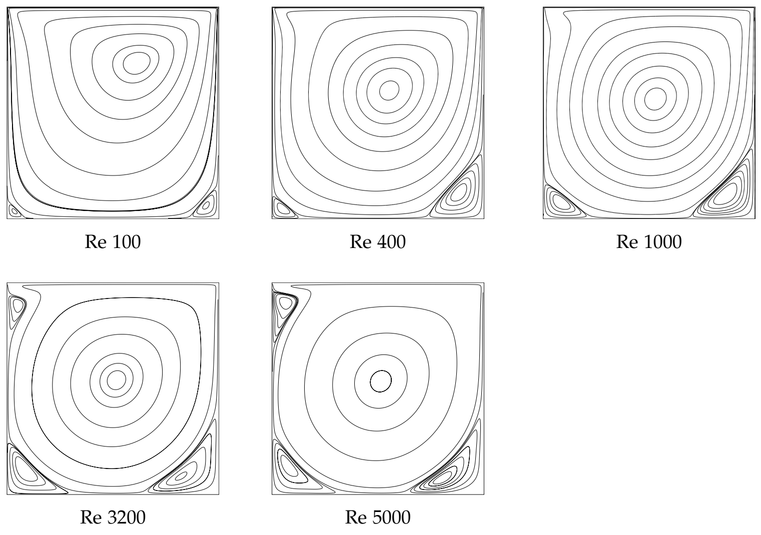


Figure 6.4.: Streamlines for two-dimensional square lid-driven cavities at different Reynolds numbers

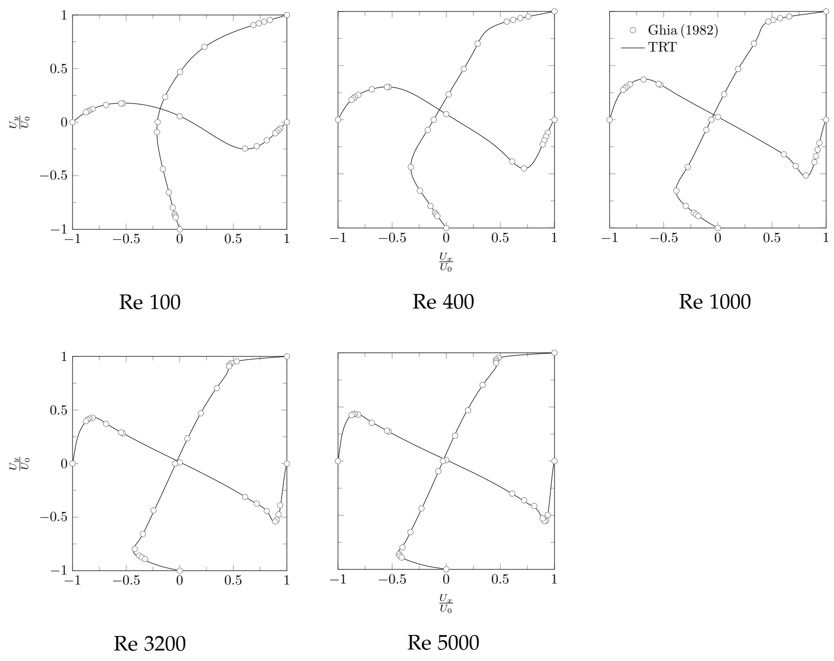


Figure 6.5.: Velocity cross-sections through the geometric centre of square lid-driven cavities at different Reynolds numbers

6.1.2. Stability of collision operators

The stability of the two collision operators, BGK and TRT, is also examined using a two-dimensional lid driven cavity: Simulations for the same domain, with two

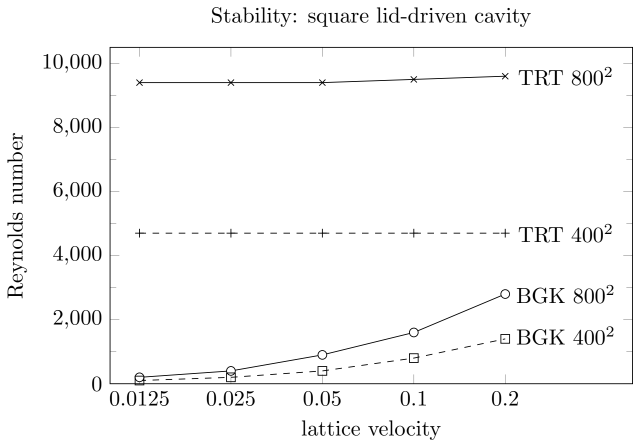


Figure 6.6.: Stability of BGK and TRT collision operators using a simple lid-driven cavity

different resolutions, 400 and 800 nodes for each spatial direction respectively, are run for different Reynolds numbers. If the simulation does not crash within $20000/U$ time steps, it is considered stable. The Reynolds number is varied for a constant characteristic velocity in steps of 50 and the highest stable Reynolds numbers for different characteristic velocities on the same grid are connected (figure 6.6). While two collision operators yield practically identical results at a similar computational cost if resolved adequately, the *TRT* collision operator is significantly *more stable* than the much simpler BGK operator: For the same grid *BGK* allows only smaller Reynolds numbers to be simulated as the *stability* is *highly viscosity dependent*. A turbulence model on the other hand increases the simulation significantly: For the case of the lid-driven cavity no instable parameter combination could be found.

6.1.3. Test case 2: Flow around a cylinder

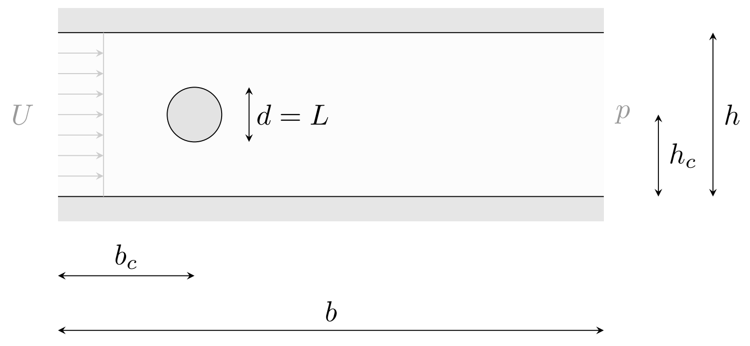


Figure 6.7.: Relevant dimensions for the flow around a cylinder

For *unsteady flow* a simple flow around a single cylinder is probably the most popular benchmark scenario. Additional to a variety of experimental results there exist CFD based results with a variety of different boundary conditions. In this thesis the inlet is chosen as a velocity boundary condition whereas at the outlet a constant pressure is enforced, which in LBM is equal to a density boundary condition (figure 6.7).

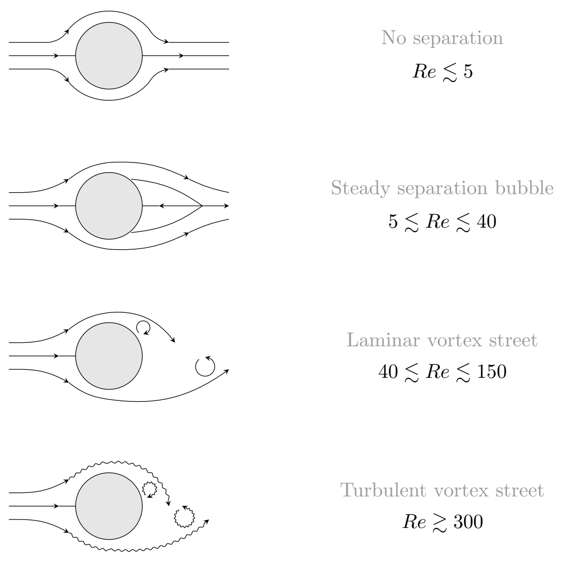


Figure 6.8.: Different flow regimes for the flow around a cylinder

While for very small Reynolds numbers (Re ≤ 5, often referred to as *Stokes flow*), the streamlines are completely symmetrical and no separation occurs, higher Reynolds numbers ($5 \leq$ Re ≤ 40) lead to the formation of *steady separation bubbles* down-stream behind the cylinder. For Reynolds numbers higher than 40 the separation becomes unsteady, resulting in vortices with alternating rotational direction, also referred to as *Kármán vortex street* with an oscillating velocity and pressure distribution (figure 6.8).[88]
The frequency of this oscillating flow can be determined by analysing the lift C_L or drag coefficient C_D (which has double the frequency of the former, appendix B.10) using an FFT analysis (appendix I.4.3.1). The characteristic number for oscillating flow is the *Strouhal number* (appendix B.11)

$$St := \frac{f\,L}{U}.$$

Through experiments the relation between Reynolds and Strouhal number as given by figure 6.9 can be found for circular cylinders. As can be seen *vortex shedding* does not happen at a single frequency but rather over a *narrow range of frequencies* leading to two limiting curves depending on the roughness of the cylinder. [79], [80]
Choosing an adequate simulation domain is though not as trivial as it might seem. If the cylinder is placed too close to the inlet or outlet, this would not lead to a proper vortex shedding. Periodic boundary conditions on the top and bottom would affect the vortex shedding frequency the least but for larger Reynolds numbers the vortex shedding overlays with the LBM pressure waves: At the pressure outlet not all populations are able to exit the domain but are reflected backwards instead. The whole domain starts acting like a huge spring and the drag coefficient fluctuates around the correct mean value but super-imposes with those pressure waves leading

[88]Generally the turbulent domain is further split up but this is not needed for the sake of this thesis.

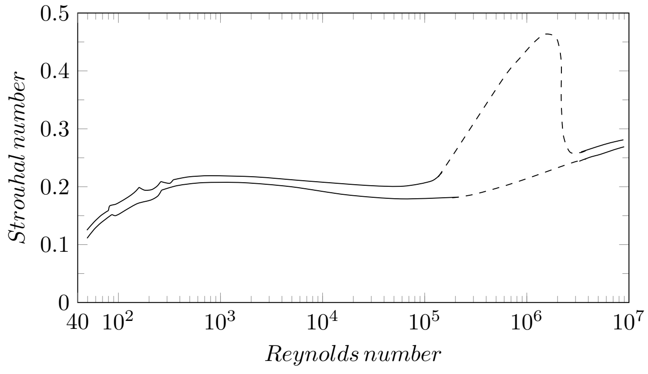


Figure 6.9.: Relation between Reynolds and Strouhal number for circular cylinders [79], [80]

to unrealistically high amplitudes and harmonic multiples of the drag force frequency, distorting the signal. Therefore for the calculation of the vortex shedding frequency and further the *Strouhal number* the lift coefficient is chosen instead of the drag coefficient, which would be more likely to be influenced by the pressure waves due to its small magnitude.

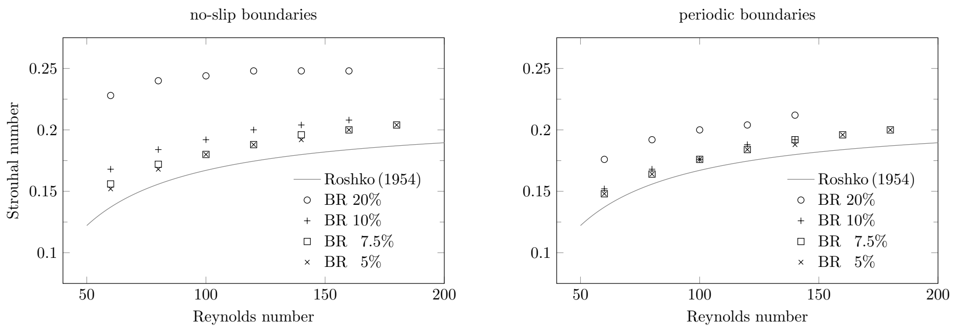


Figure 6.10.: Influence of blockage ratio on Strouhal number for simulation domains with periodic boundary conditions and restricting walls

This can be avoided by replacing the periodic boundaries with walls, although if located too close to the cylinder the ground effect leads to unrealistically high Strouhal numbers (figure 6.10) while inducing an increased pressure loss and therefore a slightly higher drag coefficient. This correlation is closely connected to the blockage ratio $BR = \frac{L}{h}$ and was studied further in [81]. The wall with its boundary layer can act as a dampener and reduce the effects slightly. Else one might use lower Reynolds numbers and velocities for a particular domain resolution or try to implement non-reflective boundary conditions. Simulation domains with periodic boundary conditions need significantly longer to stir vortex shedding but one may trigger it using small asymmetries or additional perturbations of the velocity field.
Similarly also the pressure around the cylinder (compare to figure 6.13) exhibits oscillating behaviour (figure 6.11). Therefore the pressure coefficient for every single cell has to be averaged over several time steps: As this results in a symmetrical curve generally only the left side is given in the literature.

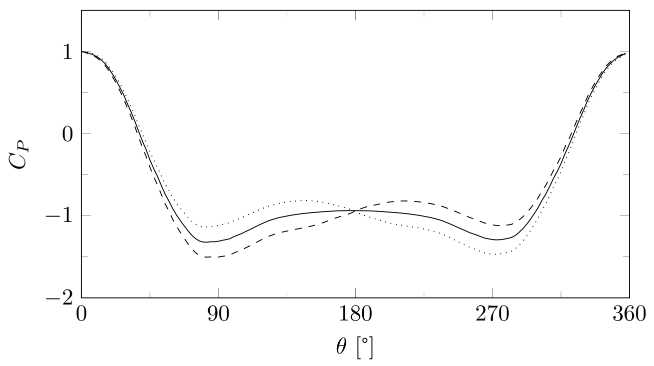


Figure 6.11.: Pressure distribution around a cylinder at Re 100 for two different time steps (dashed and dotted) and the average value (solid line)

The *flow around a cylinder* (figure 6.12) is simulated at different Reynolds numbers without and with the *Smagorinsky turbulence model* ($C_S = 0.15$). The characteristic number for oscillating flow, the *Strouhal number*, is compared to the literature as well as the *pressure distribution* (figure 6.15) and the *drag and lift coefficient* (figure 6.16). For all simulations a comparably small characteristic velocity of $U = 0.05$ is chosen whereas the domain is simulated with 400^2 or 800^2 cells with a blockage ratio of 7.5% and the inlet is chosen as $1/3$ of the domain width.

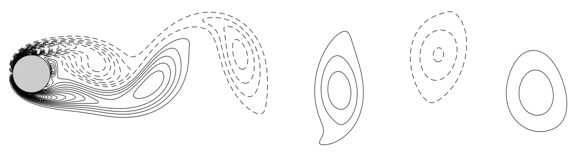


Figure 6.12.: Vorticity contours for the flow around a cylinder at Re 100

	St	$\overline{C}_D$	ΔC_D	ΔC_L
Re 100	0.1840	1.58	0.05	0.37

Table 6.1.: Strouhal number St, mean drag coefficient $\overline{C}_D$ and amplitude of drag (ΔC_D) and lift coefficient (ΔC_L) for the three-dimensional flow around a cylinder

The values for the mean drag coefficient $\overline{C}_D$ and the the corresponding amplitudes of drag ΔC_D and lift coefficient ΔC_L can be found for 2D in table 6.2 and for 3D in table 6.1. Figure 6.17 shows the vorticity of a three-dimensional simulation of the flow around a cylinder using the TRT collision operator combined with a Smagorinsky large-eddy turbulence model.[89]

[89]In this case only the magnitude of vorticity is used which is no good criterion for vortex detection as

	St	$\overline{C}_D$	ΔC_D	ΔC_L
Re 100	0.1820	1.60	0.04	0.38
Re 1000	0.2420	1.70	0.27	1.56

Table 6.2.: Strouhal number St, mean drag coefficient $\overline{C}_D$ and amplitude of drag (ΔC_D) and lift coefficient (ΔC_L) for the two-dimensional flow around a cylinder

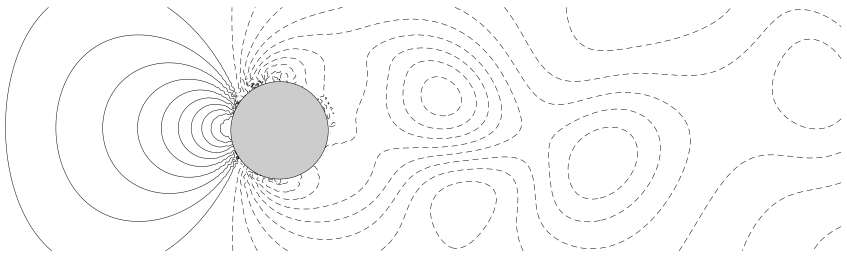


Figure 6.13.: Pressure contours around a cylinder at Re 100

The Strouhal numbers obtained in CFD are generally *higher* than the values found in experiments [82] which is also the case for our simulation (figure 6.14). Additionally it should be considered that the resolution of the FFT is limited by the lowest frequency that can be detected, the *Nyquist frequency* (appendix I.4.3.1) given by $\frac{1}{N}$. This leads to a resolution limit of the Strouhal number in LBM units for a given number of samples N according to

$$St = \frac{fL}{U}\left(\pm\frac{L}{NU}\right).$$

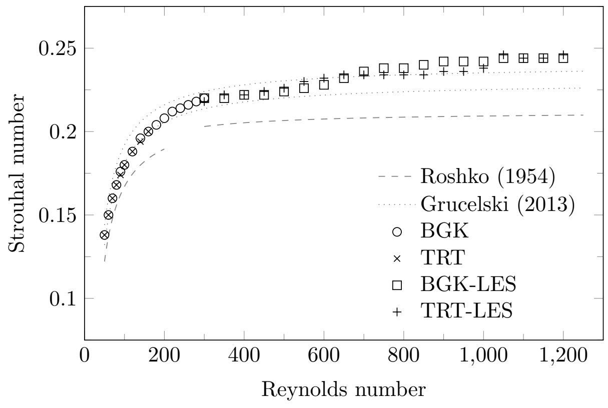


Figure 6.14.: Strouhal number for different Reynolds numbers obtained by 2D LBM simulation with and without turbulence model compared to the experimental results of Roshko [82] and the results obtained by another LBM simulation [83]

A few hundred thousand time steps might sound like a lot but depending on the characteristic length and the characteristic velocity chosen in LBM this might lead to huge uncertainties. In all the simulations in this thesis the uncertainty due to the signal resolution is kept at ±2% of the measuring interval.

described in appendix B.1. This picture should rather only schematically illustrate the presence of vortices in the flow.

Overall the results correspond well with the literature. [83]–[86] Only the pressure distribution around the sphere (figure 6.15) shows a noteworthy deviation from the benchmark solutions.

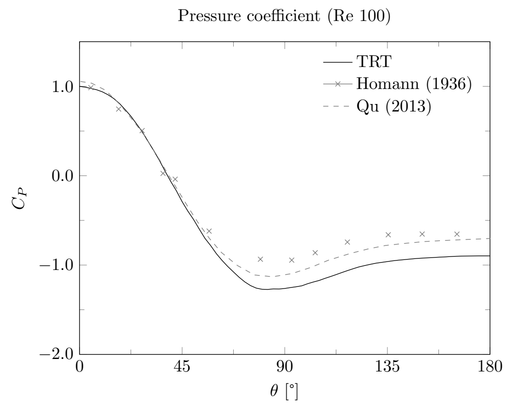


Figure 6.15.: Pressure coefficient for Reynolds 100 obtained by a 2D simulation compared to experimental values [84] and results from a finite volume code [86]

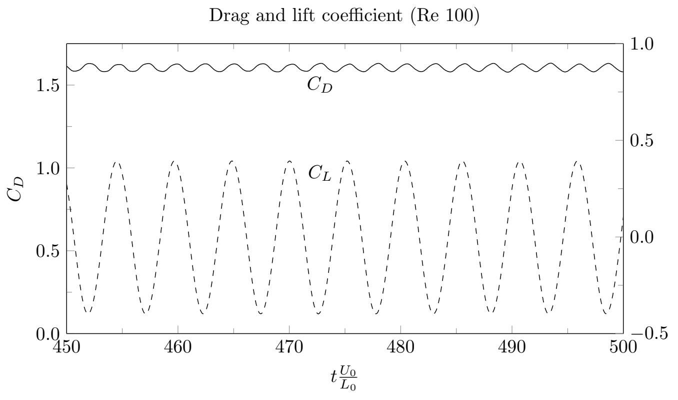


Figure 6.16.: Drag and lift coefficient for Reynolds 100 over dimensionless time

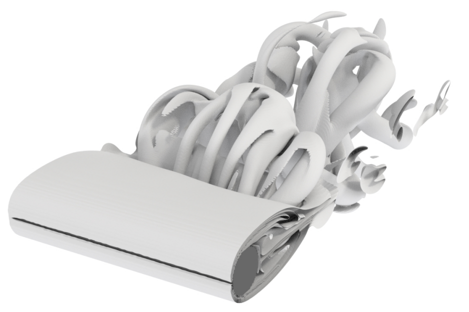


Figure 6.17.: Magnitude of vorticity around a 3D cylinder for Re 400 obtained with a TRT-LES simulation

6.1.4. Multi-component flow

A second lattice for advection-diffusion of a scalar quantity, in this case the mass-fraction, is introduced and a Gaussian hill advection in 2D and 3D is simulated. The results for advection and diffusion dominated flow are compared to the analytical solution. The Dirichlet and Neumann boundary conditions are implemented using anti-bounce-back while solid walls are modelled by half-way bounce-back. From now on only collision operators without turbulence model are considered.

6.1.4.1. Test case 3: Gaussian hill

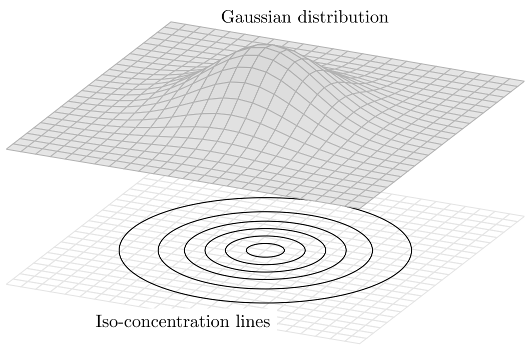


Figure 6.18.: Schematic domain of a Gaussian hill advection in 2D: The surface varies in height with the concentration while the circles underneath are corresponding iso-lines of concentration

In a Gaussian hill advection a domain is initialised with a *Gaussian distribution* (figure 6.18) of a given mass-fraction Y_0 as[90]

$$Y(\vec{x}, t = 0) = \frac{Y_0}{(2\,\pi\,\sigma_0^2)^{\frac{d}{2}}}\, e^{-\frac{(\vec{x}-\vec{x}_0)^2}{2\sigma_0^2}}$$

that evens out due to diffusion with the diffusion coefficient D and is advected with a certain velocity $\vec{u}$. The analytical solution for an indefinitely large domain can be found to

$$Y(\vec{x}, t) = \frac{1}{\sigma_0^2 + 2\,D\,t}\,\frac{Y_0}{(2\,\pi)^{\frac{d}{2}}}\, e^{-\frac{(\vec{x}-\vec{x}_0-\vec{u}t)^2}{2(\sigma_0^2+2\,D\,t)}}\,.$$

For a simulation the boundaries have to be set to periodic boundary conditions while the domain has to be chosen large enough so the boundaries do not have an impact on the solution itself. For validation purposes then the iso-concentration lines are compared to the analytical solution after a particular time step.
The precise benchmark scenarios are chosen as proposed by Krüger [5]: First only diffusion is simulated and later an advection dominated scenario by setting the Péclet number accordingly. The three-dimensional model is verified using cross-sections of a symmetric Gaussian hill which is omitted here as it is virtually indistinguishable from its two-dimensional counterpart.

[90]The mass fraction of the second component at any point in time may be calculated by $1 - Y$.

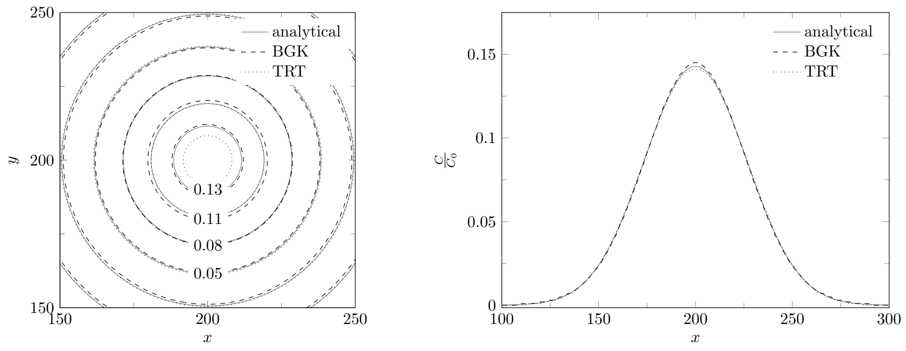


Figure 6.19.: Gaussian Hill diffusion benchmark scenario after 200 dimensionless time steps with $D = 1.5$ and $\vec{U} = (0, 0)$: Iso-lines of concentration (left) and the distribution (right)

For the *pure diffusion* benchmark a very large dimensionless diffusion coefficient $D^* = 1.5$ is chosen whereas the advection velocity is set to 0. A comparably large domain of 512^2 nodes is then simulated with a BGK and a TRT ($\Lambda = 1/6$ for lowest diffusivity error) collision operator, both with non-linear equilibrium distribution, and compared to the analytical result after 200 LBM time steps (figure 6.19). Similarly in the second scenario the diffusion is kept to a minimum with a comparably small diffusion coefficient of $D^* = 0.0043$ but now the *advection* velocity is chosen as $\vec{U} = (0.1, 0.1)$. Again the simulation is performed with both collision operators but in this case Λ is set to $1/12$ for the smallest possible advection error (figure 6.20). Both results are virtually identical with those in [5]. Even though both collision operators lead to excellent results in all further simulations TRT will be used due to its better stability properties.

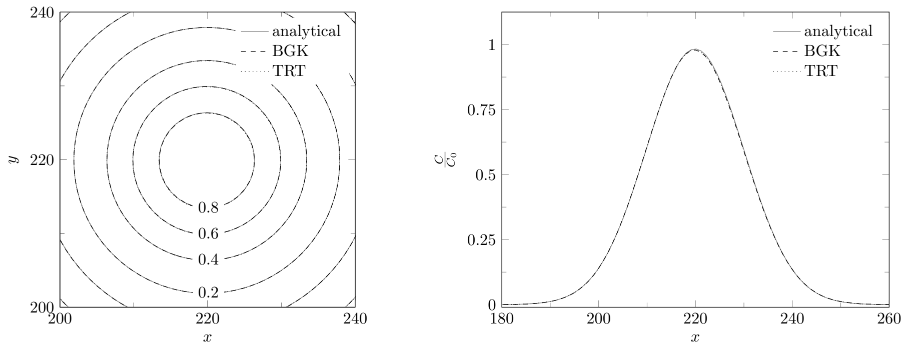


Figure 6.20.: Gaussian Hill advection benchmark scenario after 200 dimensionless time steps with $D = 0.0043$ and $\vec{U} = (0.1, 0.1)$: Iso-lines of concentration (left) and the distribution (right)

6.2. Performance

As one of the main advantage of LBM over conventional CFD lies in the computational efficiency, proper attention has to be paid to an efficient implementation. The algorithm itself is ideal for parallelisation as it only relies on local information but reaching an acceptable parallel scaling on a system with limited bandwidth poses several challenges. The following section outlines some of the issues encountered in the course of the thesis and will describe the proposed solutions.[91]

6.2.1. Benchmark system

The programming was performed on my private system with an LGA2011 socket Intel i7-4820K and dual channel memory which was later replaced by a i7-4930K six-core processor of quad channel memory with higher bandwidth. Most benchmarks were performed on a high-performance single-CPU system with the same i7-4930K processor but a smaller bandwidth. Finally a few test were run on a significantly more powerful system with a LGA2066 twelve-core Intel i9-7920X and high performance DDR4 memory in order to further investigate the parallel scaling. Table 6.3 shows the precise simulation systems, all running either Windows 7 or 8. If not mentioned otherwise the results correspond to benchmark system number three.
As also suggested by other LBM solvers like OpenLB, it is highly recommended to *turn hyper-threading* (appendix G.1.5.1) *off* for optimal performance else this might lead to very inconsistent results.

	Processor				*Memory*	
Model	*Cores (Threads)*	*Speed (Turbo)*	*L3 Cache*	*Size*	*Speed*	*Channels*
i7-4820K	4 (8)	3.7 (3.9) GHz	10 MB	16 GB	2133 MHz	2 (4)
i7-4930K	6 (12)	3.4 (3.9) GHz	12 MB	32 GB	2400 MHz	4 (4)
i7-4930K	6 (12)	3.4 (3.9) GHz	12 MB	64 GB	1333 MHz	4 (4)
i9-7920X	12 (24)	2.9 (4.3) GHz	16.5 MB	64 GB	3000 MHz	4 (4)

Table 6.3.: A list of all tested benchmark systems: Hyper-threading was disabled for all benchmark simulations.

6.2.2. Matlab

As a first prototype a basic *D2Q9* and *D3Q19 Matlab* model with *BGK and TRT collision operators* is implemented. The computational domain can be imported using the *Matlab image toolbox*: Pictures that fulfil a certain form are rescaled and split up according to the RGB channels[92] in order to identify the different boundary conditions and their corresponding orientation. The aforementioned *Zou/He pressure and velocity boundaries* are implemented as well as *full- and half-way bounce-back* no-slip walls.
The code is completely *vectorised*, making use of optimised internal functions like `circshift` for streaming and avoiding loops wherever possible. This allows Matlab to use its own optimised multi-threaded operations: The code is running on six different threads but yet, due to the interpreted implementation (appendix G.2), the performance is comparably slow with a maximum of 4.1 million lattice updates per second in 2D and as the matrix operations get increasingly more difficult with larger domains also performance drops significantly to a bit more than 1 million nodes per second (figure 6.29).

[91] An extensive programming guide for C++ and OpenMP is given in the appendix G.

[92] I recommend using uncompressed *.tiff-images as else the native image compression might render the pictures useless.

6.2.3. C++

In the next step a C++ version is programmed in order to speed-up the calculations. At first an object-oriented approach (appendix G.4.2) is taken, supposed to increase flexibility, but as the performance of the first prototype turns out to be rather disappointing, reaching not more than 7 Mlups with the *D2Q9* model on a single core, the code is rewritten using functions and optimised for *multi-threading using OpenMP* (appendix G.4.9). The following section describes the programming-technical aspects of the simulation and further investigates the parallel scalability.

6.2.3.1. Containers and memory layout

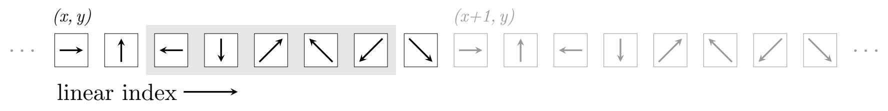


Figure 6.21.: The memory layout used for this thesis: Populations of a particular node are saved contiguous in memory

In the standard LBM algorithm the collision step is fully explicit and easy to calculate. The performance mainly depends on how fast values can be read from and written to the main memory (RAM). This means values have to be organised properly in the RAM in order to make full use of pre-fetching (appendix G.1.4): The values should be arranged in memory in such a fashion that each cache line loads only values into the cache that will be used in the next calculation step. A bad memory layout alone may lead to a performance 30 times worse than a good one. [5]

```
#include <omp.h>

#pragma omp parallel for default(none) shared(...) schedule(static)
for(unsigned int y = 0; y < NY; ++y)
{
  for(unsigned int x = 0; x < NX; ++x)
  {
    ...   //do something
  }
}
```

Figure 6.22.: The corresponding simple parallel loops: Each core handles a single cell at one time

In a linear storage like the RAM every array (appendix G.4.4) has to be laid out *linearly*: Every multidimensional array in C++ is laid out such that elements belonging to the same row are contiguous in memory (*row-major*). Similarly values in our simulation are stored in *linear arrays* in such a way that populations belonging to the same node are contiguous in memory and populations of cells sharing the same x-coordinate lay next to each other (figure 6.21). This requires a looping mechanism according to figure 6.22.[93]
In this implementation *two separate containers for a single population* are chosen: Collision and streaming is handled by a single loop and populations collide and stream from one

[93]This means in this particular implementation the main direction of the domain should coincide with the y-axis in order to yield the best possible performance. This loop is already parallelised using OpenMP.

array into the other. This way both, pre- and post-collision populations, can be *accessed separately*, which allows boundary conditions to be handled independently after the combined collision and streaming step. As the rest node does not require a streaming step it will be further isolated in an own container. Due to their enormous but yet constant size all populations are manually allocated in the heap (appendix G.1.1) using arrays while the indices of boundary cells are stored in vectors (figure 6.23).[94] In order to avoid a potential integer overflow every index as well as the corresponding loop counters for linear population indexing have to be chosen as `size_t` (defined in `<stdlib.h>`), an unsigned integer type that depending on the implementation can be anything between a `unsigned char` and an `unsigned long long`.

```
#include <stdlib.h>
#include <vector>

double *f0  = (double*) malloc(mem_size_scalar);
double *f1  = (double*) malloc(mem_size_speeds);
double *f2  = (double*) malloc(mem_size_speeds);

std::vector<size_t> wall, inlet, outlet;
wall.reserve(NX*NY*NZ);
inlet.reserve(NX*NY*NZ);
outlet.reserve(NX*NY*NZ);

...   //calculations

free(f0);
free(f1);
free(f2);
```

Figure 6.23.: Schematic memory allocation of the populations (`f0` for rest node and the two populations `f1` and `f2` for the other directions) as fixed-size double-precision arrays whereas the boundary conditions (`wall`, `inlet` and `outlet`) are variable-sized vectors

6.2.3.2. Collision and streaming

The visual approach of splitting up the algorithm into a collision

$$f_\alpha^t(\vec{x}, t+\Delta t) = f_\alpha(\vec{x}, t) + \frac{1}{\tau}(f_\alpha^{(eq)} - f_\alpha)$$

and a streaming step

$$f_\alpha(\vec{x} + \vec{e}_\alpha \Delta t, t+\Delta t) = f_\alpha^t(\vec{x}, t+\Delta t).$$

may aid understanding of the algorithm but requires two loops. Performance improves drastically (almost by a factor of two) if the *collision and streaming step* can be *combined* in a single loop instead of being performed separately.

$$f_\alpha(\vec{x} + \vec{e}_\alpha \Delta t, t+\Delta t) = f_\alpha(\vec{x}, t) + \frac{1}{\tau}(f_\alpha^{(eq)}(\vec{x}, t) - f_\alpha(\vec{x}, t))$$

At the end of the combined collision and streaming step the two population pointers pointing to the heap are swapped and boundary conditions can be applied. *Half-way bounce-back* can be implemented quite easily afterwards by reversing the populations

[94]They might change in size over time.

in the boundary cells and streaming them back to the cells they came from.
Additionally the equation should be rewritten as

$$f_\alpha(\vec{x} + \vec{e}_\alpha \Delta t, t + \Delta t) = \left(1 - \frac{1}{\tau}\right) f_\alpha(\vec{x}, t) + \frac{1}{\tau} f_\alpha^{(eq)}(\vec{x}, t)$$

introducing two *constants* $C_1 = 1 - \frac{1}{\tau}$ and $C_2 = \frac{1}{\tau}$ which can be *calculated once* at the start of the simulation and re-used for the case of constant relaxation times.

6.2.3.3. Large domains

The virtual address space (appendix G.1.2) of 32bit applications is limited to $4\,GB$, practically even lower. This means that for large domains the simulation application has to be compiled for *64bit*. Under Windows this requires a 64bit GCC distribution (for example the compiler suite TDM-GCC) or equivalent. [95]

6.2.3.4. Boundary conditions

Boundary conditions are applied after the combined collision and streaming step using the two pre- and post-collision populations. In order to yield maximum performance the boundary conditions are *hard coded* for every possible direction instead of using look-up tables.

6.2.3.5. Force calculation

While in general it is more efficient to hard-code everything instead of creating look-up tables for the force calculation in 2D and especially in 3D there are too many possible boundary node orientations to hard code them separately. Instead an algorithm should identify the solid cells that are in contact with the fluid and the corresponding populations coming from those fluid cells, assigning them bool true and false values. Then the force on a structure can be calculated using the populations after streaming and the boolean values.

6.2.3.6. Single-core performance and simple parallelisation

The following performance tests are performed using lid-driven cavities of aspect ratio one in $2D$ as well as in $3D$. The values mentioned for multi-core performance correspond to a full processor with six cores (benchmark system three).
The single core performance of the obtained code is already significantly higher than the Matlab code running on all cores as can be seen in figure 6.24, where the lower data sets correspond to the single core performance. The D3Q19 simulation with more than double the speeds and thus double the calculation steps is about half as fast. No matter the lattice the performance of TRT is about 20% slower than BGK whereas the

[95] After installation the compiler has to be set up properly in Code::Blocks: Under compiler settings a new compiler has to be configured, the compiler flags added and the linker settings (include OpenMP static library) as well as the toolchain executables have to be adjusted accordingly. Finally the compiler has to be set for the particular project individually. Do not forget to activate the option 'append settings...' in the Code:Blocks project options else the compiler flags might not be applied.

collision operators with turbulence models are again 40% slower: The computation of the local strain tensor at every time step makes this algorithm significantly more computationally heavy. Compared to traditional RANS-based CFD this is though a comparably small loss in performance.

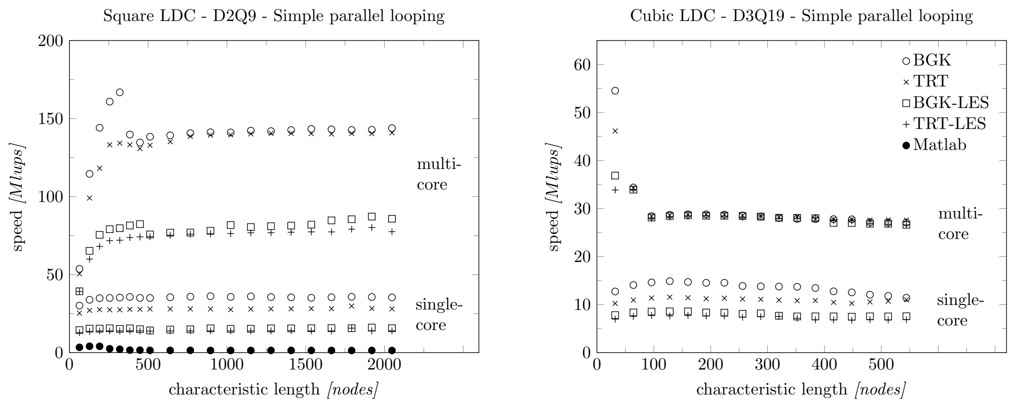


Figure 6.24.: Scalability of performance across the domain size (left: 2D, right: 3D) for the case of simple parallel loops. Matlab is natively parallelised on all six cores while for the C++ implementations the lower data set corresponds to a single core and the upper one corresponds to a full processor.

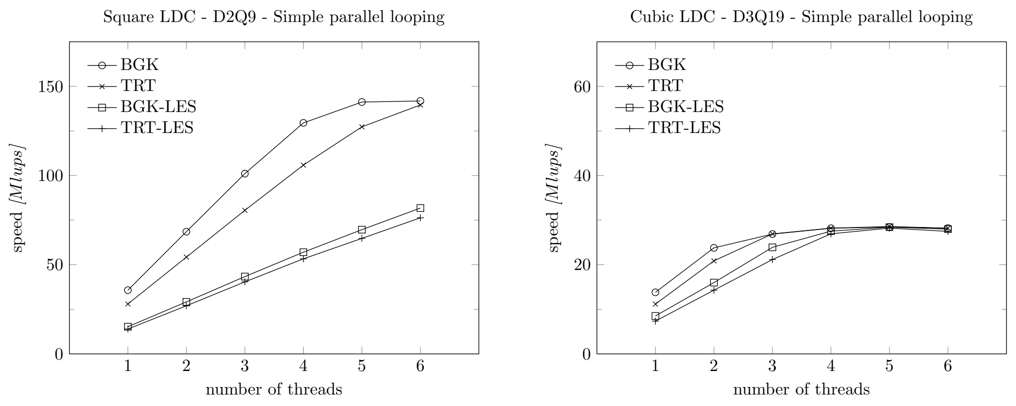


Figure 6.25.: Speed-up across the number of threads for the simple parallel loops for different collision operators and a domain of 256^2 (left) and 256^3 nodes (right) respectively. The bandwidth seems to limit the scalability as too many values have to be loaded again from memory.

When parallelising the code, using the simple parallel looping as depicted in figure 6.22, for very small 2D domains the overhead introduced by OpenMP is significant and the scaling over the number of threads is quite poor, larger two-dimensional domains *scale almost linearly* underlining LBMs parallel potential (figure 6.25). Compared to Matlab the multi-threaded 2D implementation is *up to 100 times faster* depending on the domain size (figure 6.24). Parallelising the 3D code by simply creating more threads that handle a single cell leads though to a severe issue: Not all cores can be used to their full potential. After creating four threads in 2D and two in 3D, additional threads do not lead to a performance increase. This is probably due to bandwidth limitations as another benchmark on my private system seems to show. As for low domain size the required values are still likely already loaded into the cache due to the comparably small number of cells for larger domains the performance quickly falls off. In 3D, where there are significantly more cells this is more visible whereas in 2D this barely has an effect. This can also be seen in figure 6.25: All collision operators are limited by a certain constant number of loaded values. Therefore for optimal parallelisation it has to be avoided at all cost to reload values to cache multiple times, instead all values loaded to the cache should be re-used as often as possible (*cache locality*).

6.2.3.7. Increasing cache locality in loops

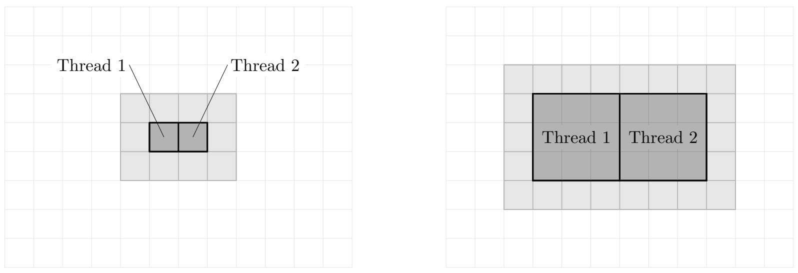


Figure 6.26.: Simple parallel looping (left) compared to parallel looping in blocks (right): For the simple parallel loop neighbouring nodes are handled by separate threads (dark gray) that require information from their next neighbours (light gray) which most are not available in memory and have to be loaded. In the case of looping in blocks each thread is assigned multiple nodes and therefore the ratio between nodes that are loaded in memory already and nodes that have to be loaded again is significantly better.

While the combination of memory layout and loops leads to an almost linear parallel scalability for 2D simulations it suffers from scaling-issues in 3D as the wrong values are pre-fetched with increasing domain size and the bandwidth is too small to load them again (figure 6.26). In order to increase locality the 3D domain is further *decomposed into cubes of a fixed size that are handled by a single thread* (figure 6.27).[96] This requires a structure of nested loop running over the different blocks and the corresponding x, y and z coordinates (figure 6.28). For the used processor the sweet spot was found to a cube size of 32.

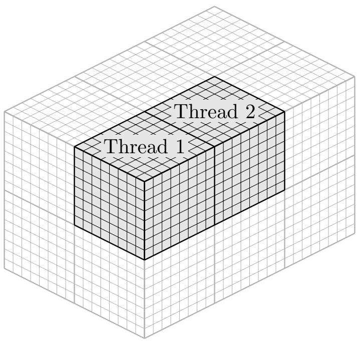


Figure 6.27.: Domain decomposition into blocks that are handled by a single thread

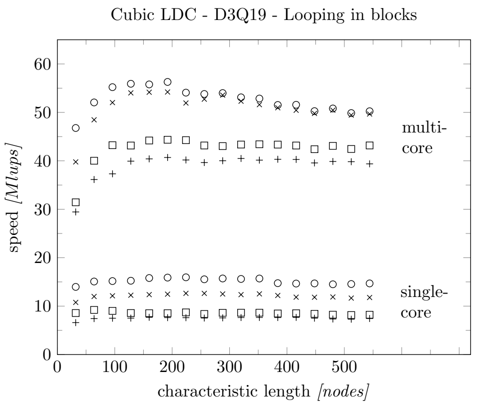
[96]All credits for this technique go to StackOverflow user Dominique LaSalle, PhD.

```
#include <omp.h>
#include <cmath>
#include <algorithm>

const unsigned int BLOCK_SIZE = 32;
const unsigned int numBlocksZ = std::ceil(static_cast<double>(NZ) / BLOCK_SIZE);
const unsigned int numBlocksY = std::ceil(static_cast<double>(NY) / BLOCK_SIZE);
const unsigned int numBlocksX = std::ceil(static_cast<double>(NX) / BLOCK_SIZE);
const unsigned int  numBlocks = numBlocksX*numBlocksY*numBlocksZ;

#pragma omp parallel for default(none) shared(...) schedule(static)
for(unsigned int block = 0; block < numBlocks; ++block)
{
  unsigned int startZ = BLOCK_SIZE * (block / (numBlocksX*numBlocksY));
  unsigned int   endZ = std::min(startZ + BLOCK_SIZE, NZ);
  for(unsigned int z = startZ; z < endZ; ++z)
  {
    unsigned int startY = BLOCK_SIZE*((block % (numBlocksX*numBlocksY)) / numBlocksX);
    unsigned int   endY = std::min(startY + BLOCK_SIZE, NY);
    for(unsigned int y = startY; y < endY; ++y)
    {
      unsigned int startX = BLOCK_SIZE*(block % numBlocksX);
      unsigned int   endX = std::min(startX + BLOCK_SIZE, NX);
      for(unsigned int x = startX; x < endX; ++x)
      {
        ...   //do something
      }
    }
  }
}
```

Figure 6.28.: Required parallel looping structure for a domain decomposed into blocks



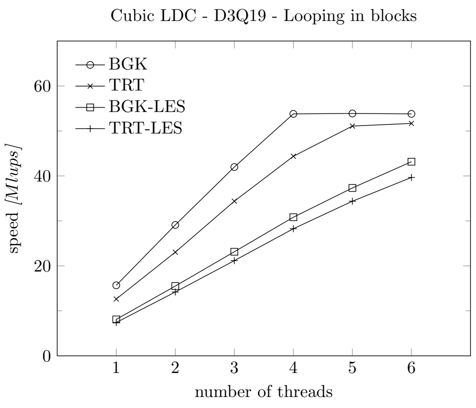
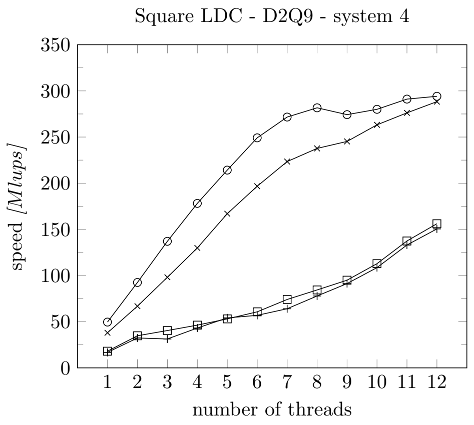


Figure 6.29.: Performance and speed-up for a three-dimensional lid driven cavity where each parallel loop handles an entire block of the domain instead of a single node

Even though this implementation scales significantly better with this technique, it starts to deviate from linear scalability for more than four cores. Again the *bottleneck* seems to be the *memory bandwidth*: Due to usage of two different arrays for the pre- and post-collion values populations twice as many values have to be loaded.[97] Arranging the cubes additionally in another cube structure using a look-up table boosts the performance slightly for large domains but leads to way more complicated addressing and looping. In order to reduce this bottle-neck one might try to additionally arrange the values in memory conformingly in cubes or choose a memory layout where the two populations co-exist in the same array and each pre-collision population is followed

[97]As we will see in the next section other implementations seem to use a different approach.

by its post-collision population. The results suggest though that for modern 2666MHz DDR4 modules the code could maintain almost linear scaling for up to 8 cores for BGK and TRT collision operators and for more than 12 cores in the case of the models with Smagorinsky turbulence model.



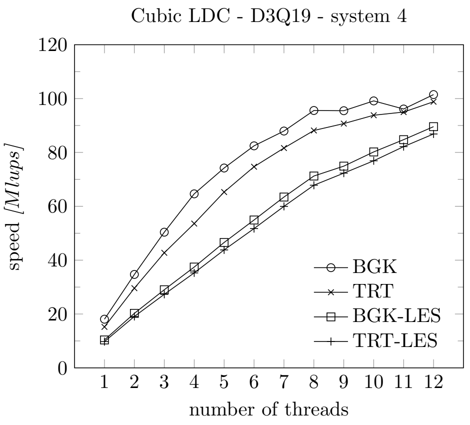


Figure 6.30.: Speed-up for a two- and three-dimensional lid driven cavity for the benchmark system 4 with twelve cores: Again the bandwidth limits the maximum number of lattice updates per second

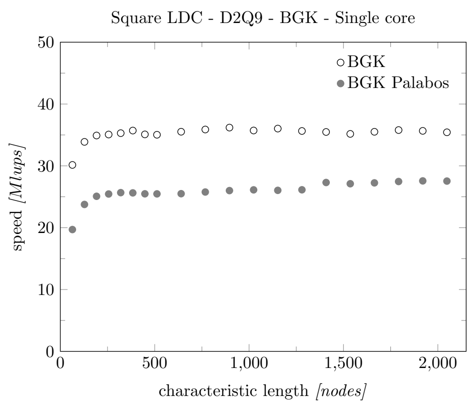
Therefore the code is run on a powerful twelve-core machine with a large bandwidth (benchmark system 4). Again for the primitive collision operators and more than eight cores the code suffers from the same issue as with the six-core benchmark systems (figure 6.30): The corresponding cores do not lead to an increase in performance. The collision operators with turbulence model on the other hand scale almost linearly. Interestingly the slope of the three-dimensional speed-up curve changes with the numer of cores. It can be shown to precisely correlate with a feature present on Intel processors, the turbo boost technology (appendix G.1.3): When more cores are active the clock count of all cores is reduced. According to Wikichip[87] this happens for this particular processor with three, five and nine cores.

6.2.3.8. Boundary conditions

Boundary conditions generally act locally on populations and can be calculated quite fastly. Therefore they are highly inefficient to parallelise: The overhead introduced by parallelisation outweighs its benefits by far in particular in 2D. Therefore the 2D simulation is significantly slower due to this sequential step. The performance can be improved though by *loop unrolling* (section G.4.6) or forcing a thread to calculate multiple adjacent boundary cells by setting the parallel for schedule to `(static, n)` where n is the number of cells one thread should calculate. While in 2D in my simulations this lead to a slight performance decrease, for 3D it gives an enormous performance boost. The speed of the 2D simulation is significantly impacted by this sequential task slowing down the scaling. The corresponding impact can be estimated using Amdahl's law (appendix G.4.8).

6.2.3.9. Comparison to Palabos: Single-core performance and memory consumption

In order to emphasise the efficiency of the presented implementation, the code is compared to the open source implementation *Palabos* in terms of single-core performance. Palabos is an open source implementation of the Lattice-Boltzmann method, developed by FlowKit Ltd. in close cooperation with the University of Geneva, that may be used on single computers or even entire clusters using OpenMPI.
Figure 6.31 shows that our optimised code is even slightly better in terms of single core performance than Palabos.[98]



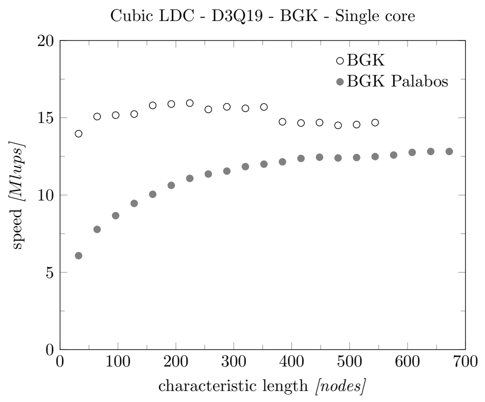


Figure 6.31.: The proposed code compared to Palabos in terms of single core performance

As can be seen from the memory consumption (figure 6.32) Palabos takes though a different approach. Only one population is used and therefore almost only half the hard drive space is required. An efficient implementation becomes though significantly more difficult: The collision and streaming is handled using a single population but in order to not overwrite values that are needed later on, additional buffers have to be introduced. The main advantages of this approach are obvious: Less bandwidth is needed and domains with double as much memory consumption and hence double as many nodes may be simulated on a system with limited memory.

6.2.3.10. Comparison to Palabos: Multi-core performance and scalability

Sadly Palabos is based on OpenMPI which is no longer officially released for Windows systems and therefore a direct comparison is not accessible for our particular Windows setup. For this reason we will take the results officially released by Palabos that were obtained on multi-processor server systems running Linux with lower clock speeds and *scale* the performance according to the ratio of *clock rate*. This should give us quite an accurate clue of how the MPI code would perform on our system. The official benchmarks furthermore show not only the accuracy of this simple scaling approach but also suggest that this slightly overestimates the real world performance.[99]

[98]Palabos is compiled using the supplied benchmark scenarios with the `-O3` compiler flag.
[99]The relevant setups are based on various Xeon processors X5570 (2.93GHz), X5550 (2.66GHz) and E5500 (2.26GHz) that all support triple channel memory and up to 1333MHz bandwidth. The clock count

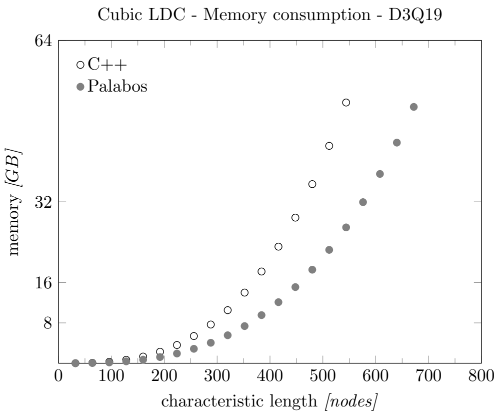


Figure 6.32.: Memory consumption of the proposed implementation compared to Palabos: The memory consumption of the proposed implementation is almost twice as high for the same amount of nodes due to the two used populations

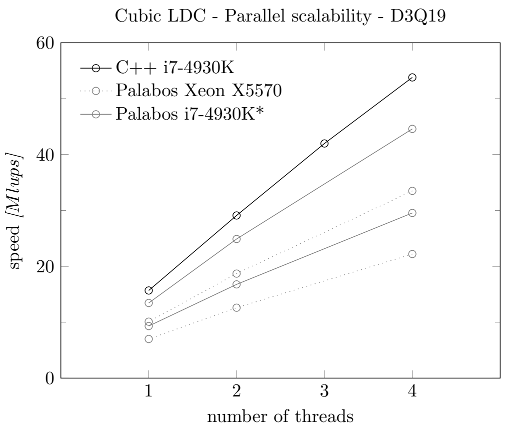


Figure 6.33.: Parallel scaling of our implementation compared to the data provided by Palabos for a different processor of slower clock count (gray dotted) and an optimistic estimate for the used processor for our benchmarks (gray solid). The two lines correspond to two different domain sizes: 101^3 (lower) and 401^3 nodes (upper).

In figure 6.33 the official results given by Palabos for the best multi-processor system, a Xeon X5570 with 2.93GHz, with a BGK collision operator and a D3Q19 lattice for two domain sizes of 101 (lower dotted curve) and 401 (upper dotted curve) are rescaled with the clock rate ration $3.9/2.93 = 1.33$ and plotted against the results obtained by our simulation. As can be seen the *performance obtained with Palabos* is estimated to be significantly *inferior* for small domain sizes and slightly inferior for large domains. The parallel efficiency (appendix G.4.7) of the BGK operator implemented in Palabos is around 83% which is slightly slower than the 86% obtained with this code. The collision operators with turbulence model scale even stronger with a parallel scalability of up to 95% (figure 6.29).

ratios of the slower two to the one with the highest frequency are given by 0.91 and 0.77 whereas the corresponding performance ratios are given by $0.89 - 1.05$ and $0.73 - 0.87$.

6.3. Porous bed - comparison to Ansys Fluent

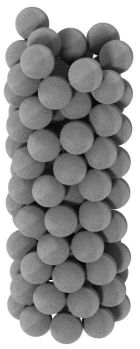


Figure 6.34.: Bed made of random spheres

In the next step additional boundaries for the mass-factions are included and a coupled fluid flow and advection-diffusion transport simulation is compared to standard CFD for the case of a flow through a porous bed made of random spheres.

6.3.1. Geometry generation and simulation setup

An artificial cylindrical domain with a diameter of 15mm is filled with 105 spheres, with a diameter of 4mm each, using the Packed Bed Generator PBG V.2 for Blender by B. Partopour and A.G. Dixon in order to obtain a realistic porous bed.[100] The coordinates of the centre of the spheres can be exported to a text file and directly imported into the LBM simulation while the Fluent mesh generation is significantly more complicated.
We evaluate the obtained 3D model with a step tracer experiment: A non-disturbed flow of a single species ($Y = 0$) is suddenly rinsed with another species ($Y = 1$). For the mass-fractions at the inlet a Dirichlet boundary condition with a given mass fraction is imposed, the walls are assumed impermeable and the outlet is modelled by a zero-flux condition. For the fluid flow we stick to a velocity inlet and at pressure outlet just like for the previous simulations and the in- and outlet is chosen to one third of the porous bed height.

6.3.2. LBM: mesh generation and settings

In order to evaluate the required simulation time a couple of 2D cross-sections are simulated and the corresponding cumulative residence time distribution is tracked. The 2D cross-sections exhibit enormous domain contractions and dead waters and therefore give rather a pessimistic view of the real three-dimensional domain.[101] The simulation run-time until a stationary flow field can be assumed is estimated to two

[100]The corresponding domain was created by a former graduand, Markus Pieber, in the course of his closely related master thesis. Thanks a lot!
[101]I suggest this approach should be taken in general when doing more time consuming three-dimensional simulations. It is a really good way of getting a pessimistic estimate.

thirds of a second while the cumulative RTD should be close enough to unity already after 3.5 seconds. In particular it must be noted that the tested 2D cross-sections due to the enormous contraction have huge overspeeds and the characteristic velocity has to be adjusted accordingly to deliver accurate results that are not plagued by huge compressibility errors.
For the real three-dimensional simulation we first calculate the transient flow field until a stationary flow can be assumed. Then the step experiment is started and the hydrodynamic and species transport equations are solved at the same time. In traditional CFD it is common practice for stationary systems with a passively transported scalar with one-way coupling to calculate the stationary velocity field and then only solve the scalar transport equation, which is referred to as *frozen flow field*. We take a similar approach in a second simulation: We iterate the velocity field with LBM until there is no visible macroscopic change and we then only run the mass-fraction transport equation. This effectively cuts the simulation time for the advection-diffusion step in half as the algorithm has only to be executed for the advection-diffusion population and not for the hydrodynamic population as well.
The precise simulation settings can be seen in table 6.4. This is equal to a simulation with a Schmidt number of $Sc = 1$ and a characteristic particle Reynolds number of $Re_p = 10$ or an inlet Reynolds number in terms of the diameter of the tube of $Re = 37.5$.

	Physical units	*LBM units*
Density ρ	$1.138\,kg/m^3$	1
Characteristic length L	$0.015\,m$	146
Characteristic velocity U	$0.036533\,m/s$	0.005
Kinematic viscosity ν	$1.46132 \cdot 10^{-5}\,m^2/s$	0.01947
Diffusivity D	$1.46132 \cdot 10^{-5}\,m^2/s$	0.01947
Hydrodynamic runtime	$1.05\,s$	75 000
Advection-diffusion runtime	$3.52\,s$	250 000

Table 6.4.: Porous bed: Physical and simulation parameters

When setting up the simulation proper attention has to be paid to specifying the simulation correctly: Small changes in simulation parameters might lead to hugely different results as described in appendix H.2.

6.3.3. Fluent: mesh generation and simulation setup

The following mesh generation and Fluent simulation was done by Markus Pieber in terms of his closely related master thesis about multi-component flow in complex geometries with Ansys Fluent.
In order to use the generated geometry for a Fluent simulation it has to be further modified creating bridges between adjacent spheres and walls using a corresponding Python script in Blender. The resulting Java script code must be run in the Ansys design modeller, creating the corresponding geometry. The final domain is then obtained by generating a tube and applying a Boolean subtraction of the tube and the spheres.
Two thirds of inlet and outlet are meshed using the sweep method whereas the rest is meshed using tetrahedral elements with prisms in the two inflation layers assigned to the walls. The resulting mesh with 23 772 604 cells and a maximum skewness of 0.86 is imported into Fluent, smoothed in order to improve cell quality before being

converted to polyhedra followed by another smoothing step. The final mesh consists of 7 555 102 cells.
The steady state is initialised using 1000 iterations while the pressure at the inlet, mass flow at the outlet and the velocities at two monitor points in front and behind the bed are tracked. With this resulting steady-state flow field the transient transport equation for a user-defined scalar is solved using the PISO algorithm with a temporal resolution of $0.001s$ (overall 4000 time steps) while the flow field is treated as frozen.

6.3.4. Porosity and tortuosity

The porosity of the domain is in this particular case not easily defined: The top of the domain is not completely packed: The bottom is pretty dense with a $\phi = 0.45$ while the top is pretty loose and therefore depending on the chosen domain this leads to porosities of $\phi = 0.45 - 0.51$. The tortuosity is estimated using the single-component LBM flow simulation and averaging the ratio between streamlines and porous domain length using 150 samples. The corresponding results show good agreeance with common values found in the literature (figure 6.5): The slightly shorter streamlines and therefore the slightly lower tortuosity are probably caused by the confining walls.

Method	*Author*	*Formula*	*Tortuosity*
LBM		CFD	1.20
Spheres	Bear and Bachmat $\phi = 0.48$ [88]	$\tau = \sqrt{\frac{2\phi}{3(1-1.209(1-\phi)^{\frac{2}{3}})}}$	1.21
	Bear and Bachmat $\phi = 0.45$ [88]	$\tau = \sqrt{\frac{2\phi}{3(1-1.209(1-\phi)^{\frac{2}{3}})}}$	1.27

Table 6.5.: Estimation of the tortuosity: Our results compared to suggestions found in the literature for random regular spheres

6.3.5. Pressure drop

The presence of the porous medium in the channel introduces a pressure drop that in this particular simulation would be too small to be determined experimentally but it can be compared to the Ansys Fluent CFD simulation and the empirically derived formulas presented in chapter 5. When comparing the results to Ansys (table 6.6) we can again see that *LBM slightly overestimates* the tiny *pressure* drop. The experimentally derived formulas lead to even lower values for $\phi = 0.48$ but to values closer to Fluent if a lower porosity of $\phi = 0.45$ is chosen. When calculating the pressure drop by taking into account that the domain is made up by 90% of the dense $\phi = 0.45$ packaging and only 10% is packed loosely this leads to a pressure drop of around $0.9 Pa$.[102]

[102]The pressure drop in the tube itself, given by the law of Hagen-Poiseuille $\lambda = \frac{64}{Re}$ and the law of Darcy-Weisbach $\Delta p = \frac{\rho U^2}{2} \lambda \frac{l}{d}$, is more than an order of magnitude smaller and can be neglected.

Method	*Pressure drop* Δp $\left[Pa = \frac{N}{m^2}\right]$
LBM	1.08
Ansys Fluent	0.98
Kozeny-Carman equation $\phi = 0.48$	0.67
Ergun equation $\phi = 0.48$	0.68
Kozeny-Carman equation $\phi = 0.45$	0.91
Ergun equation $\phi = 0.45$	0.92

Table 6.6.: The pressure drop obtained by LBM compared to Ansys Fluent and the two empirical formulas

6.3.6. Cumulative residence time distribution

We calculate the cumulative residence time assuming that the mass-fractions of the pseudo-incompressible LBM simulation correspond to a comparable incompressible simulation. Therefore the density and thus also the volumetric flow rate at in- and outlet can be assumed to be equal ($\rho = \rho^{(in)} = \rho^{(out)}$ and $\dot{V} = \dot{V}^{(in)} = \dot{V}^{(out)}$) and the cumulated residence time distribution can be calculated from the discrete cells at the in- and outlet according to

$$F = \frac{c_i^{(out)}}{c_i^{(in)}} = \frac{\frac{\dot{n}_i^{(out)}}{\dot{V}^{(out)}}}{\frac{\dot{n}_i^{(in)}}{\dot{V}^{(in)}}} = \frac{\dot{m}_i^{(out)}}{\dot{m}_i^{(in)}} = \frac{\sum Y_i^{(out)} \rho^{(out)} u_n^{(out)}}{\sum Y_i^{(in)} \rho^{(in)} u_n^{(in)}} = \frac{\sum Y_i^{(out)} u_n^{(out)}}{\sum Y_i^{(in)} u_n^{(in)}}$$

where u_n is the velocity normal to the outlet. To be precise only the mass fractions entering the domain at the inlet and leaving it at the outlet should be considered but we will simply assume that any back-flow will lead to a corresponding forward flow due to the incompressibility assumption and the difference in concentration may be neglected.
The resulting residence time distribution is in excellent agreeance with the Ansys simulation (figure 6.35): The deviation between the two curves is barely visible. The corresponding temporal evolution of the iso-surface Y = 0.6 obtained by LBM can be seen in figure 6.36.

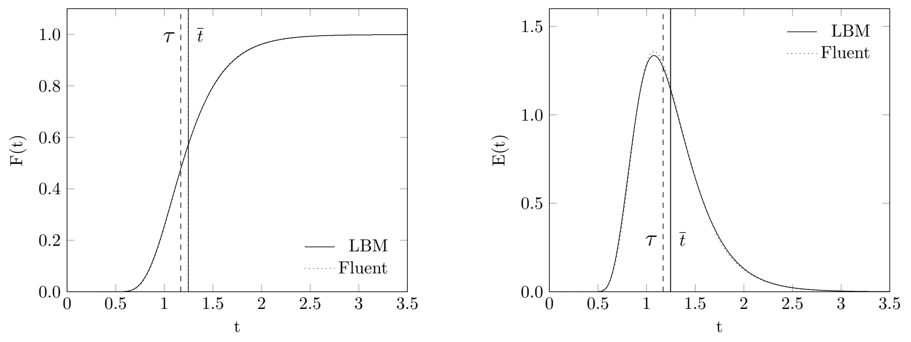


Figure 6.35.: Residence time distribution (left) and its cumulative counterpart (right) obtained by LBM and Ansys Fluent

The average residence time can be estimated by numerical integration to

$$\bar{t} = \int_{t=0}^{\infty} t\, E(t)\, dt \approx \sum_i t_i\, E(t_i) = 1.2473s$$

whereas the Fluent simulation using far less discrete time steps for the integral approximation yields 1.2454s. The hydrodynamic dwell time for this particular case is given by

$$\tau = \frac{V}{\dot{V}^{(in)}} = \frac{\phi \frac{d^2\pi}{4} H}{\frac{d^2\pi}{4} U} = \frac{\phi H}{U} = \frac{7740877}{7740877 + 3427635} \cdot \frac{600}{0.005} = 83171.80 = 1.1695s.$$

The difference between the hydrodynamic dwell time and average residence time is most likely caused by *dead waters* that lead to an additional contraction of the effective flow path.

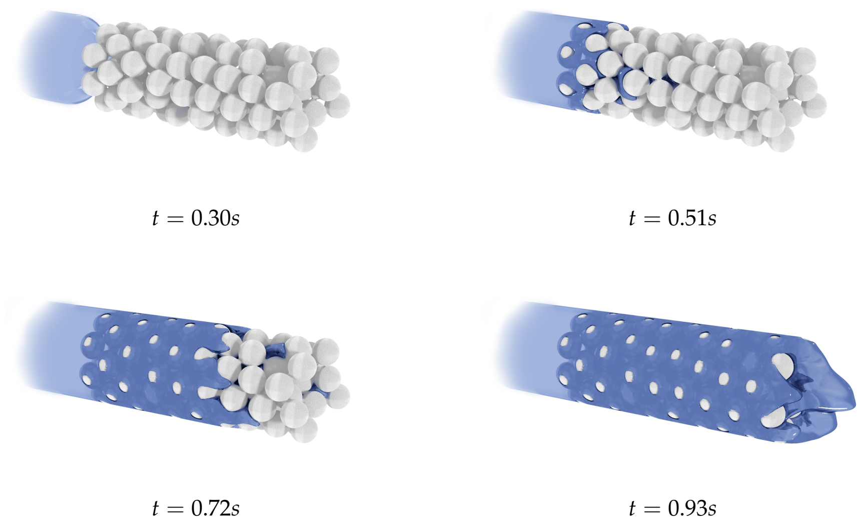


Figure 6.36.: Iso-surface of the mass-fraction Y = 0.6 at different time steps obtained by the LBM simulation

The distribution fulfils its fundamental property

$$\int_{t=0}^{3.75} E(t)dt = F(t = 3.75) = 1.0002$$

which furthermore undermines the correctness of the proposed simulation.

6.3.7. Simulation runtime

Table 6.7 gives a short overview of the settings and runtime of the three simulations on comparable six-core systems (similar to benchmark system number 3).

	LBM transient	*LBM frozen flow*	*Fluent frozen flow*
Total number of cells	15.01 M	15.01 M	7.55 M
Number of relevant cells	8.74 M	8.74 M	7.55 M
Mesh generation	less than 20 s	less than 20 s	several hours
Hydrodynamic flow runtime	3.87 h	3.87 h	2.45 h
Hydrodynamic time steps	75 000	75 000	1 000
Advection-Diffusion runtime	28.80 h	14.45 h	9.70 h
AD time steps	250 000	250 000	4 000

Table 6.7.: Comparison of the runtime of LBM (transient and frozen flow field) and Ansys

It must be mentioned that the corresponding simulation runtimes may only act as *guidelines*. In Fluent it might be possible to resolve the domain with less cells, apply a greater time step and a less strict convergence criterion. At the same time also in LBM higher characteristic velocities (the results suggest that the simulation time could be cut in half by doubling the characteristic velocity without breaching the incompressibility assumption) for a more coarse temporal resolution but increasing the compressibility error and a more coarse spatial resolution could be chosen.
For stationary flow with a frozen flow field Ansys Fluent beats the LBM simulation: LBM has to transiently iterate the flow field until no change on macroscopic level is visible. Furthermore it is restricted to a very small time step due to the highly intricate geometry.

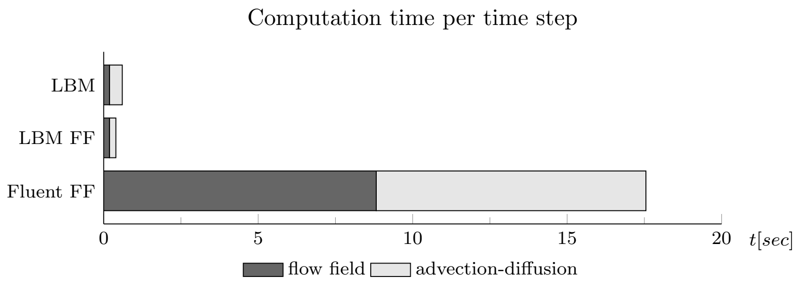


Figure 6.37.: Comparison of the simulation runtime per time step of the different methods: LBM is roughly 60 times faster per time step for both the stationary flow field as well as the advection-diffusion equation.

On the other hand *LBM* is *virtually meshless*: There is a regular grid but the corresponding geometry can be generated in significantly less than a minute even for huge three-dimensional domains taking up several dozens gigabytes of hard drive space as compared to the difficult and time-consuming mesh generation in Fluent taking at least several hours. Additionally it must kept in mind that uses 62.5 as many time steps with a significantly higher temporal resolution (figure 6.37). Last but not least for a *transient* simulation Fluent would take significantly longer than Lattice-Boltzmann: LBM has apply a simple algorithm for two populations, the hydrodynamic population f and the advection-diffusion population g, whereas in traditional CFD the continuity equation as well as the momentum equation for all three spatial directions have to be explicitly satisfied. A completely transient simulation of the entire domain with Ansys would most likely take about a week.

7. Application: Real porous media

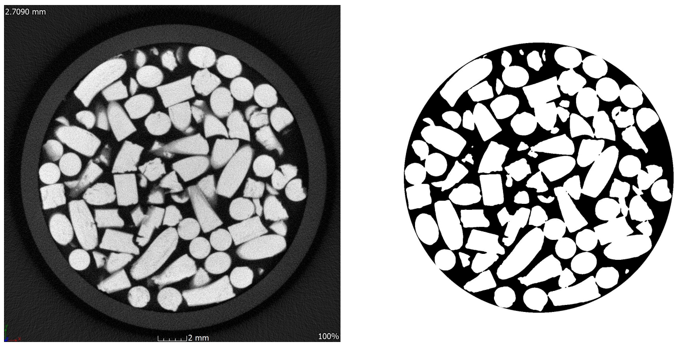


Figure 7.1.: Micro-CT scan and enhanced computational domain: Raw scan (left) and processed image (right)

In the final step the validated model is applied to a binary multi-component flow in realistic porous media in order to underline the advantages of LBM over traditional methods in complex geometries. Meshing such a highly intricate domain in traditional CFD is almost impossible.

7.1. Computational domain

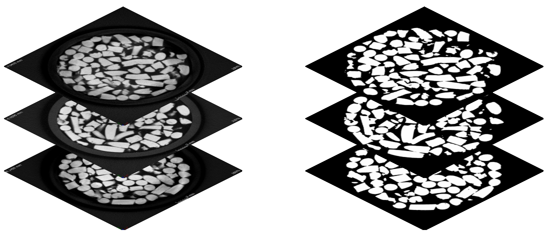


Figure 7.2.: Grid for a porous medium (right) created from X-Ray Micro-CT (left)

Using X-Ray Micro-CT, cross-sections of a porous absorber is obtained. Meshing the resulting geometry for traditional CFD is not viable but using the corresponding domain in a Lattice-Boltzmann simulation by *stair-casing* the boundary is feasible. The cross-sections are imported into Matlab, the contrast is enhanced turning the picture into a binary image and finally the domain is rescaled with a basic *nearest-neighbour interpolation*. In order to avoid non-sense pixels due to the present grain a small-scale *Gaussian blur* is applied (figures 7.1 and 7.2). This logical domain can be stored and imported into the C++ simulation (figure 7.3). The high resolution of the first hydrodynamic simulation with *90 million cells* comes though at a cost of memory space: The hydrodynamic simulation takes up almost $30 GB$ of memory.

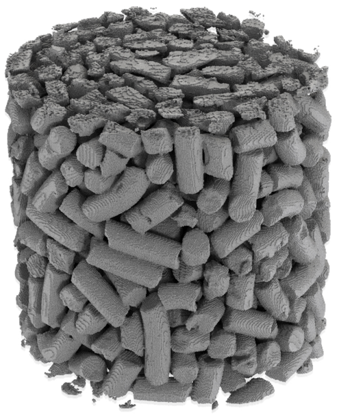


Figure 7.3.: Computational grid created from CT cross-sections

7.2. Simulation setup

The *simulation parameters* are chosen somewhat *arbitrarily*. In the future this simulation will be benchmarked against experiments but for now only arbitrary yet realistic values are used in order to illustrate the potential of the method without being able to discuss its accuracy or even verify its validity.
The Reynolds number at the inlet is chosen identical to the simulation of the bed with spheres, $Re = 37.5$. Similarly also the Schmidt number is set to unity and the LBM characteristic velocity $U = 0.005$ remains unchanged. The domain is though resolved significantly more accurate: For the first hydrodynamic simulation a highly accurate domain of 90 million nodes is used but for the simulation involving the step experiment a significantly lower resolution of 34 million nodes is chosen. This delivers optimal performance without taking up too much memory: The simulation could be easily performed on a $32\,GB$ system.

7.3. Results

The simulation runtime for $1.7\,s$ of hydrodynamic flow amounts to $30\,h$ of simulation time and the following $2.5\,s$ of advection-diffusion take around $40\,h$ on benchmark system three. The obtained results are discussed in the following section.

7.3.1. Porosity, tortuosity and diffusion coefficient

The porosity is calculated counting the corresponding fluid cells and dividing them by the overall volume of the sample. This leads to a value of $\phi = 0.41$.

A simple single-component simulation for low Reynolds number flow is carried out and using 150 streamlines (figure 7.4) the tortuosity is estimated by the ratio of total streamline length to length of the computational domain. The value is compared to values found in the literature for random porous media and regular spheres. As can be seen in table 7.1 the tortuosity found due to the LBM simulation is significantly lower than the ones given in the literature for random porous media and is even slightly lower than values found for random spheres. This is attributed to the huge channels present in this particular porous medium and the confining walls that only slightly redirect the flow.

Method	*Author*	*Formula*	*Tortuosity*
LBM		CFD	1.18
Spheres	Bear and Bachmat [88]	$\tau = \sqrt{\frac{2\phi}{3(1-1.209(1-\phi)^{\frac{2}{3}})}}$	1.35
	Zalc et al. [89]	CFD	1.44
Porous	Koponen et al. [90]	$\tau = 1 + 0.8(1-\phi)$	1.47
	Comiti and Renaud [91]	$\tau = 1 + 0.63\, ln(\frac{1}{\phi})$	1.56
	Yu and Li [92]	$\tau = \frac{1}{2}\left(1 + \frac{1}{2}C + C\frac{\sqrt{(\frac{1}{C}-1)^2+\frac{1}{4}}}{1-C}\right)$ with $C = \sqrt{1-\phi}$	1.66

Table 7.1.: Estimation of the tortuosity: Our results compared to suggestions found in the literature for random regular spheres and random porous media for $\phi = 0.41$

7.3.2. Pressure drop

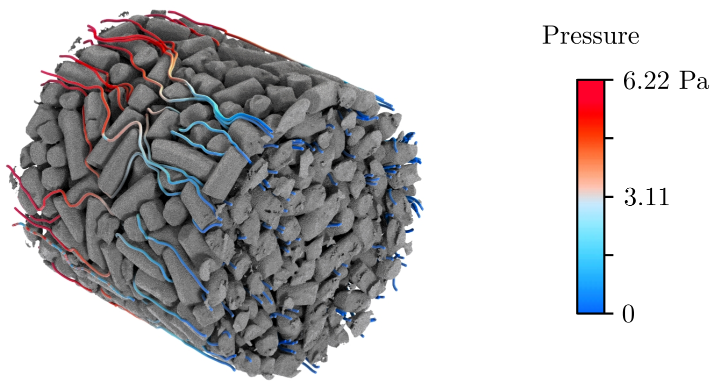


Figure 7.4.: Streamlines of the single-component simulation in porous media coloured according to the pressure drop

The overall pressure drop is determined to $6.22 Pa$ which is significantly higher than suggested by the empiric formulas that propose values just slightly under $2 Pa$ respectively. Streamlines for the corresponding simulation coloured according to the pressure drop are depicted in figure 7.4.

7.3.3. Residence time distribution

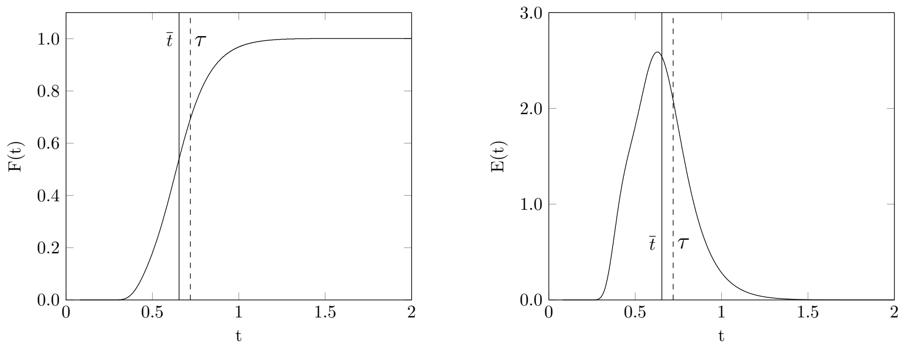


Figure 7.5.: Residence time distribution (left) and its cumulative counterpart (right) for the real porous medium: The deformation of the distribution function is a result of the bypass visible in figure 7.6

Similarly to the porous bed of spherical particles a residence time distribution is obtained by a stationary frozen field advection-diffusion simulation (figure 7.5).

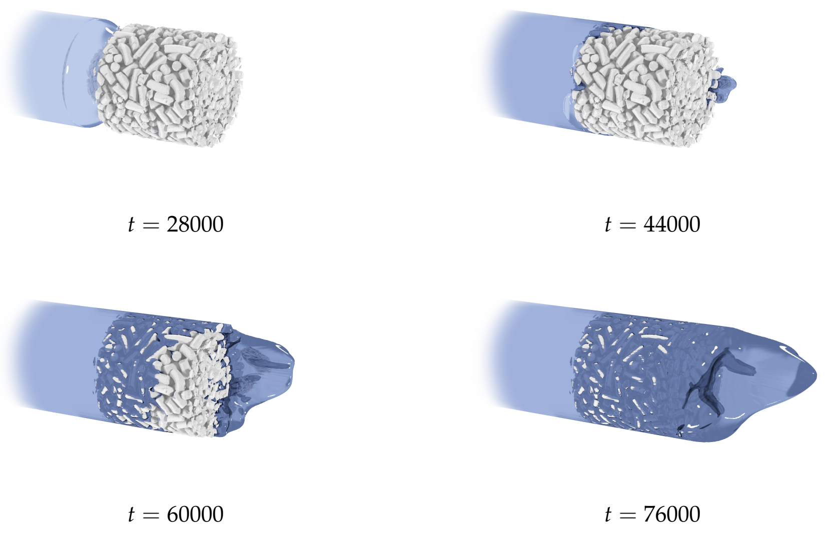


Figure 7.6.: Iso-surface of the mass-fraction Y = 0.6 at different time steps obtained by the LBM simulation

The mean residence time is calculated to

$$\bar{t} = \int_{t=0}^{\infty} t\,E(t)\,dt \approx \sum_i t_i\,E(t_i) = 0.6542s$$

and the hydrodynamic dwell time is given by

$$\tau = \frac{V}{\dot{V}^{(in)}} = \frac{\phi \frac{d^2\pi}{4} H}{\frac{d^2\pi}{4} U} = \frac{\phi H}{U} = \frac{18096022}{18096022 + 5748811} \cdot \frac{545}{0.005} = 82720.90 = 0.7196s.$$

In this case a *bypass* within the domain (figure 7.6) is visible that leads to a slightly deformed residence time and a hydrodynamic dwell time that is longer than the mean residence time.

8. Conclusion and Outlook

8.1. Conclusion

In this thesis an efficient Lattice-Boltzmann algorithm, that is able to handle fluid flow as well as the transport of a scalar through advection and diffusion, was implemented on a multi-core system using C++ and OpenMP. For this purpose *two different arrays for each population* are allocated in the heap: The population of one array collides and streams into the second array and after each iteration their two pointers are swapped. This *eliminates the usage of buffers* and allows for an easy implementation of boundary conditions as pre- and post-collision populations can be accessed separately. Within the array the populations are saved as row major such that *populations* of a particular cell are *contingent in memory* in order to make effective use of pre-caching. In order to increase cache locality a special *looping* system is introduced: Each core is given a *block* of the three-dimensional domain instead of a single cell. While this is not consistent with the memory layout it still effectively boosts the parallel performance as it reduces cache misses: Less values have to be loaded from the memory again.
In context of this thesis *four different collision operators* were implemented: The simple *BGK* operator with one relaxation time, the more complex *TRT* collision operator offering an additional freely tunable relaxation time and corresponding versions with a Smagorinsky large-eddy *turbulence model*. Implementing a TRT collision operator instead of the simple BGK model leads to a drastic increase of stability at an only 20% slower performance while turbulence models reduce the performance further by around 40%. The *parallel scaling* of all collision operators is *excellent*, reaching 86% for the more bandwidth-heavy base algorithms and almost 95% for the algorithms with turbulence models. Due to the usage of two different populations the *memory requirement* is though almost twice as high as similar implementations and therefore also the *required bandwidth* to be almost *twice* as large. Empirically a necessary bandwidth of around $1333 MHz$ per core running at around $4 GHz$ was found. If the required bandwidth is not met additional cores won't lead to a computational speed up. This means that for current high performance systems with quad channel $2666 MHz$ DDR4 memory this particular implementation is only able to make use of around 8 cores efficiently. Though if a single CPU system meets these requirements this implementation is slightly faster than the novel implementation Palabos and offers more consistent performance for different domain sizes. For a twelve-core processor system was shown to achieve 295 million lattice updates per second with the $D2Q9$ lattice and 100 million with the $D3Q19$ lattice.
The efficient model is then validated against *standard benchmark scenarios* for single component and binary multi-component flow with excellent agreeance and exceptional computational performance. Only the *pressure* was found to be *slightly higher* in some instances which is attributed to the compressible nature of Lattice-Boltzmann: As the solver is only pseudo-incompressible numerical errors might be absorbed into the density and might lead to slightly different pressure as the latter is connected to the density by the speed of sound. Due to the resulting *pressure waves* in particular

for highly turbulent flow, characterised by high Reynolds numbers, a LES turbulence model may be used successfully. In this case though special boundary conditions, less prone to reflection of pressure waves, should be implemented.
Including additional physical phenomena such as multi-component flow is straight forward using another lattice accounting for advection-diffusion of mass fractions. When compared to Ansys Fluent for a steady-state, advection-dominated species-transport simulation in a porous bed made of spheres, LBM was able to calculate the transient flow on an identical system ina similar time frame Ansys takes for a frozen flow field approach for roughly the same number of nodes. As LBM is *inherently time dependent* if used for steady state flow the algorithm has to be iterated until no visible change on macroscopic level is visible and due to the intricate geometry the temporal resolution of LBM has to be chosen very fine: If Fluent would have to the same temporal resolution it can be estimated to take 42 times longer, let alone a transient simulation. This perfectly underlines the potential of LBM for transient flow in complex porous media.

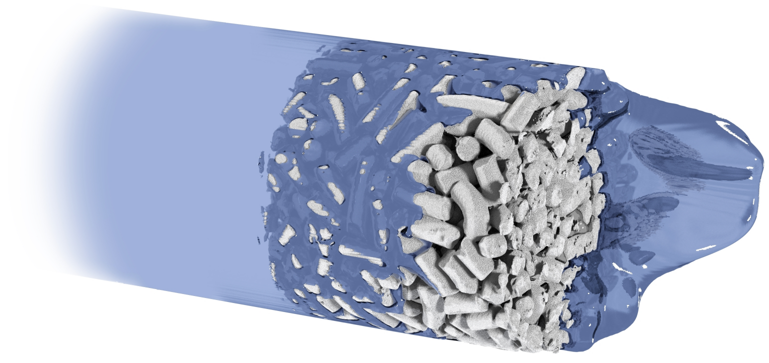


Figure 8.1.: Realistic porous medium: iso-surface of the mass fraction $Y = 0.6$ after 60000 timesteps

Finally, from *cross-sections obtained by a computed tomography* a LBM simulation domain is created and simulated. While we were not able to mesh the same domain with Ansys Fluent the grid generation with Lattice-Boltzmann can be completely automated and can be done in a matter of seconds. The results seem plausible but a comparison with extensive experimental results is needed. Further it should still be investigated if the results are also comparable for diffusion dominated flow. For this sake it might be necessary to adjust boundary conditions to account for a diffusive flux at the inlet. Further the current model may be extended in the future to account for additional, more complicated interactions or may be ported to clusters as described below.

8.2. Physical outlook: reactive multi-component flow

It seems obvious to extend the current multi-component model to account for *chemical reactions* of multiple species and include an additional lattice for the energy equation.

LBM also seems to be a promising tool to solve the radiative transfer equation, as it can be derived, similarly to hydrodynamics, from a viewpoint of collisions of light particles, so called photons. As a natural consequence one could try to couple radiation and fluid dynamics in a consistent LBM multi-physics scheme.

8.3. Computational outlook: GPGPU and MPI

As outlined in section 6.2.3.7 for the proposed code around 1300MHz bandwidth are needed in order to supply a single core with data. For modern quad-channel systems with a maximum bandwidth of slightly over 3200MHz per module the maximum number of cores on a shared memory setup with OpenMP are probably limited to 12, the rest of the cores will sit there idling, waiting for data. Moreover even Intel's recently announced high-end server platform Cascade Lake, featuring 48 core processors and twelve-channel RAM, will struggle to keep all cores busy. At the huge price-tag such a setup would come with, chances are it would be most likely still be outperformed by a single high-performance GPU implementation at a significantly lower hardware costs. This means a code parallelised on only a single processor will be somewhat unlikely to deliver significant performance boosts over the next couple of years. If more performance is needed the existing code should be either *coupled with OpenMPI* (traditional cluster) or more economically be *ported to graphics cards*. In that case the chosen code structure should be reconsidered. Probably in order to make better use of the bandwidth a more complex algorithm involving only one array for each population and a *buffer* should be chosen.

Appendix A.

Thermodynamics

The science dedicated to the *relation between temperature, heat and energy* is referred to as *thermodynamics*. It is governed by four basic principles, the four laws of thermodynamics.

A.1. State and process variables

In thermodynamics one distinguishes between two main types of quantities: *State variables* depend only on the current equilibrium state of a system and do not depend on the path it has taken to reach the corresponding condition whereas *process variables* are an adequate measure for the path taken to reach the corresponding state. In order to differentiate these two types of quantities the latter is generally denoted by small Greek delta δ instead of exact differentials d.
This means for a state variables $f(u, v)$ the order of derivatives does not matter, the symmetry of second derivatives, the *Schwarz's theorem*

$$\partial_{uv} f = \partial_{vu} f,$$

must hold.
State variables can be further classified: *Extensive* variables such as energy E or volume V depend on the size of the system whereas *intensive variables* such as pressure p or temperature T do not. Dividing an extensive variable by the mass of the corresponding system yields so called *specific variables*. In the case of the volume this leads to the specific volume v which is the reciprocal of the density ρ

$$v = \frac{V}{m} = \frac{1}{\rho}.$$

A.2. Internal energy

One main measure of the state of a thermodynamic system is the so called internal energy, an abstraction for a variety of different forms of energy such as kinetic and rotational energy of the corresponding *molecules* that reflect the *internal state* (section A.1) of a fluid regardless of the macroscopic fluid flow. The change of the internal energy can be described using temperature and volume as

$$dE_i = \left(\frac{\partial E_i}{\partial T}\right)_V dT + \left(\frac{\partial E_i}{\partial V}\right)_T dV.$$

A.3. The first law of thermodynamics for a closed system

The change of the internal energy may be described using a basic *conservation of energy* given by the *first law of thermodynamics* (figure A.1). A closed system is able to store energy either in macroscopic *external energy E_a* such as potential or kinetic energy of the fluid flow or the aforementioned *internal energy E_i* (section A.2). These quantities are state variables (section A.1) and are therefore denoted with exact differentials. For a closed system (no mass transport) the energy can only be changed by the *work* W and *heat* Q transferred to the system, which generally depend on the exact process path and therefore are denoted by partial differentials. The most common type of work is the expansion of the corresponding control volume against the surrounding pressure p that can be calculated to $\delta W_V = -p\,dV$. Therefore for a closed system the following energy budget must hold:

$$\delta Q + \delta W = dE_i + dE_a$$

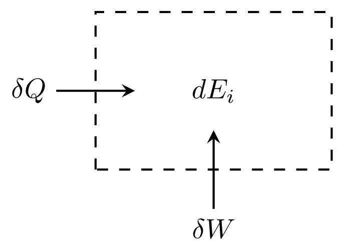


Figure A.1.: First law of thermodynamics for a stationary closed system

A.4. Ideal gas

So far no assumption regarding the material was made. However, the equations are simplified significantly when assuming gases characterized by simple interactions. One such simplified model is the ideal gas, an idealised model of a real gas where *particles of infinitesimal size interact with each other only in elastic collisions*. This allows for a simple description including a very simple equation of state given by

$$pV = R_m T.$$

Additionally it can be found experimentally (Joule expansion) that the internal energy (section A.2) of such an ideal gas is no function of the volume the gas occupies but rather only a function of the temperature .

A.4.1. Perfect gas

An even more basic behaviour can be achieved by neglecting the intermolecular forces resulting in *constant heat capacities* (section A.5). Such a model gas is referred to as a *calorically perfect gas*.

A.5. Enthalpy and heat capacities

Combining the first law of thermodynamics for a close system (section A.3) with the differential of the internal energy, neglecting changes due to the external energy, yields

$$\delta Q - p\,dV = \left(\frac{\partial E_i}{\partial T}\right)_V dT + \left(\frac{\partial E_i}{\partial V}\right)_T dV.$$

Considering an isochor system, meaning the terms including dV vanish, and dividing the equation by the mass, results in an equation of the specific heat

$$\partial q = \left(\frac{\partial e_i}{\partial T}\right)_v dT$$

where the term

$$c_v := \left(\frac{\partial e_i}{\partial T}\right)_v$$

is referred to as the *heat capacity at constant volume*.
Analogously the *enthalpy* H is introduced as

$$H := E_i + pV$$

which takes the place of the internal energy for isobar processes and leads to the *heat capacity at constant pressure*

$$c_p := \left(\frac{\partial h}{\partial T}\right)_p.$$

One can find a correlation between the two heat capacities using

$$de = c_v dT + \left(\frac{\partial e_i}{\partial v}\right)_T dv$$

which can be rewritten using the first law of thermodynamics to

$$\partial q - c_v dT = \left[\left(\frac{\partial e_i}{\partial v}\right)_T + p\right] dv$$

This is equivalent to

$$\left(\frac{\delta q}{dT}\right)_p - c_v = \left[\left(\frac{\partial e_i}{\partial v}\right)_T + p\right]\left(\frac{\partial v}{\partial T}\right)_p = c_p - c_v.$$

As for ideal gases (section A.4) the internal energy is only a function of temperature but not of the specific volume this leads to

$$c_p - c_v = R_m.$$

Additionally the *heat capacity ratio* is defined as the ratio between the two specific heats

$$\kappa := \frac{c_p}{c_v}.$$

A.6. Entropy

Using the findings from the section above we are able to formulate the specific heat in the case of a reversible process as (section A.3)

$$\delta q_{rev} := de_i + pd\hat{v} = c_V dT + pd\hat{v}.$$

If the heat was a state variable it would have to fulfil the symmetry of second order derivatives (section A.1). This is though not the case due to

$$\left(\frac{\partial c_v}{\partial \hat{v}}\right)_T \neq -\left(\frac{\partial\left(\frac{R_m T}{\hat{v}}\right)}{\partial T}\right)_{\hat{v}} = -\left(\frac{\partial p}{\partial T}\right)_{\hat{v}}.$$

As c_v (section A.5) is by definition independent of the specific volume the left side is equal to zero whereas the right side in the case of an ideal gas yields $-\frac{R_m}{\hat{v}}$. If the temperature on the right side would not appear, the Schwartz's theorem (section A.1) would be fulfilled and we would have found a state variable. This can be achieved introducing the *entropy* s, using the integrating factor $\frac{1}{T}$, which yields

$$ds := \frac{\delta q_{rev}}{T} = \frac{c_v}{T} dT + \frac{p}{T} d\hat{v}.$$

For reversible processes this entity vanishes whereas for every *irreversible* process, which is the norm for common processes found in nature, it *steadily increases* (second law of thermodynamics). Entropy is responsible for the asymmetry of physical laws in time, it gives processes direction: Differences in temperature drive every process and are the source of this irreversibility.[103] Integrating the differential definition assuming a perfect gas (section A.4) yields

$$\frac{p}{p_1} = \left(\frac{\hat{v}_1}{\hat{v}}\right)^{\kappa} e^{\frac{s-s_1}{cv}} = \left(\frac{\rho}{\rho_1}\right)^{\kappa} e^{\frac{s-s_1}{cv}} = \left(\frac{T}{T_1}\right)^{\frac{\kappa}{\kappa-1}} e^{-\frac{s-s_1}{Rm}}.$$

Which can be rewritten assuming isentropy $s - s_1 = 0$ to

$$\frac{p}{\rho^{\kappa}} = const.$$

[103] According to Boltzmann, entropy is hidden information contained in a collection of degrees of freedom which are too small to be seen and too numerous to keep track of. It is proportional to the number of states in an ensemble that satisfies a certain criterion ($S \propto log(N)$). The system will evolve towards more likely states, meaning that the entropy will tend to the maximum and will then rattle around between similar states.

Appendix B.

Fluid dynamics

B.1. Vorticity and vortex detection

The *vorticity* of a fluid field is defined as the curl of the velocity vector $\vec{u}$

$$\vec{\omega} = \nabla \times \vec{u}$$

and is a measure for the *local spin* observed by a point travelling with the flow.

B.1.1. Vortex detection

There is *no mathematical definition of a vortex*. Generally it is used to describe the rotation of coherent structures in turbulent flow around a particular point.
While for the case of 2D flow the vorticity degenerates to a scalar value which may be used for vortex detection in three dimensions this yields an entire vector. Therefore one might use different and more complex criteria to detect vortices.[104] The following section describes some widely used measures and methods.

B.1.1.1. Vorticity magnitude

The *absolute value of the vorticity*

$$|\vec{\omega}| = |\nabla \times \vec{u}|$$

might seem like a good measure for vorticity but it leads to unrealistic high values near to the walls. Similarly also the *helicity magnitude*

$$(\nabla \times \vec{u}) \cdot \vec{u}$$

might not detect vortices properly.

B.1.1.2. Negative pressure threshold

As centripetal forces lead to high velocities and therefore to *low pressure in vortices* one might simply visualise iso-surfaces of high negative pressure. While this is easy to implement and is purely local, it is also somewhat arbitrary and pressure may vary significantly along a vortex.

[104] Apart from the here mentioned local criteria there exist several more complicated methods (Banks-Singer, Sujudi-Haimes) for identifying the corresponding vortex core lines

B.1.1.3. Advanced criteria

There are several more advanced criteria available that allow for better vortex detection. Most commonly either the Q-, Delta- or Lambda2-criterion are used. All of them use some sorts of *invariants of the strain rate tensor*, generally by decomposing it into a symmetric and antisymmetric part. For accurate vortex detection one should turn to these criteria but should be at the same time be aware that they might lead to very different results.

B.2. Conservation equations in integral notation

The differential conservation equations are only valid if the velocity and density fields contain no abrupt changes. The conservation equations in integral form B.1 are *more general* also containing the solutions were this can't be assumed.[105]

$\frac{\partial}{\partial t} \int \rho \, dV = - \int \rho \vec{u} \cdot \vec{n} \, dA$	*Continuity equation*
$\frac{\partial}{\partial t} \int \rho \vec{u} \, dV = - \int (\rho \vec{u}(\vec{u} \cdot \vec{n}) + p\vec{n}) \, dA + \int \rho \vec{g} dV$	*Momentum equation*
$\frac{\partial}{\partial t} \int \rho e \, dV = - \int (\rho e(\vec{u} \cdot \vec{n}) + p\vec{u} \cdot \vec{n}) \, dA$	*Energy equation*

Table B.1.: The conservation equations in integral notation

B.3. Conservative and non-conservative form

Traditionally the conservation equation are written as a balance of fluxes, the variables of interest do not appear as coefficients, which is referred to as the conservative form. The equations can though be rewritten by applying the chain rule and eliminating corresponding terms using the other conservation equations. While this is mathematically equivalent this changes when discretising the two different forms.

B.3.1. Non-conservative momentum equation

Applying the chain rule to the left hand side of the momentum equation with the use of the continuity equation yields the non-conservative form

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{1}{\rho}\frac{\partial \tau_{ij}}{\partial x_j}.$$

[105]The corresponding discontinuities are given by the Rankine-Hugoniot relations. Therefore sometimes the integral notation is also referred to as global and the differential one as local form.

B.3.2. Non-conservative energy equation

Analogously the energy equation can be rewritten to

$$\rho\frac{\partial e}{\partial t} + \rho u_j \frac{\partial e}{\partial x_j} = \frac{\partial}{\partial x_j}(k\frac{\partial T}{\partial x_j}) + \frac{\partial}{\partial x_j}(\tau_{ij} u_i)$$

And by rewriting it introducing the specific enthalpy $h = e_i + \frac{p}{\rho}$ with

$$\rho\frac{Dh}{Dt} = \rho\frac{De_i}{Dt} + \frac{Dp}{Dt} - \frac{p}{\rho}\frac{D\rho}{Dt}$$

where the last term is equivalent to zero due to the continuity equation and combining the resulting pressure terms with the right hand side we yield

$$\rho\frac{\partial h}{\partial t} + \rho u_j \frac{\partial h}{\partial x_j} = \frac{\partial}{\partial x_j}(k\frac{\partial T}{\partial x_j}) + \frac{\partial}{\partial x_j}(\tau_{ij} u_i).$$

B.4. Other important dimensionless numbers

The other two dimensionless numbers that can be found in table 1.3 but are not further relevant in this thesis are

$Fr := \frac{U}{\sqrt{gL}}$	$\frac{\text{flow inertia}}{\text{gravity}}$	*Froude number*
$Pr := \frac{\mu c_p}{k} = \frac{\nu}{a}$	$\frac{\text{viscious diffusion rate}}{\text{thermal diffusion rate}}$	*Prandtl number*

B.5. Euler and Lagrange specification

Generally the conservation equations are formulated in a *still frame* (*Eulerian specification*). It is also possible to specify the equations *following an individual fluid parcel* (*Lagrangian specification*).[106]

B.6. Incompressible flow

B.6.1. Divergence free velocity field

Strictly speaking incompressible flow only means that a *fluid parcel is not compressed along its way on a stream line* (Lagrangian specification, section B.5). This is not equivalent to a constant density throughout the flow field but instead only requires the Lagrangian derivative of the density to vanish

$$\frac{D\rho}{Dt} = \frac{\partial\rho}{\partial t} + u_i\frac{\partial\rho}{\partial x_i} = 0.$$

[106]Even though theoretically both descriptions could be used, normally the one or the other is more suiting for a particular problem.

Applying the chain rule to the continuity equation and inserting it into the equation above yields

$$\frac{D\rho}{Dt} = -\rho\frac{\partial u_i}{\partial x_i} = 0$$

meaning that incompressible flow is equivalent to a *divergence free velocity field*.

B.6.2. Derivation of the Poisson's equation for pressure

Calculating the gradient of the momentum equation yields

$$\frac{\partial}{\partial x_i}\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial}{\partial x_i}\left\{-\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{1}{\rho}\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]\right\}$$

which can be split up into

$$\frac{\partial}{\partial x_i}\left(\frac{\partial u_i}{\partial t}\right) + \frac{\partial}{\partial x_i}\left(u_j\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial}{\partial x_i}\left(-\frac{1}{\rho}\frac{\partial p}{\partial x_i}\right) + \frac{\partial}{\partial x_i}\left\{\frac{1}{\rho}\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]\right\}$$

and simplified using the chain rule and the divergence free condition

$$\frac{\partial}{\partial x_i}\left(\frac{1}{\rho}\frac{\partial p}{\partial x_i}\right) = -\frac{\partial u_j}{\partial x_i}\frac{\partial u_i}{\partial x_j} + \frac{\partial}{\partial x_i}\left\{\frac{1}{\rho}\frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]\right\}.$$

Assuming a constant shear viscosity[107] and density this can be further simplified to

$$\frac{1}{\rho}\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial u_j}{\partial x_i}\frac{\partial u_i}{\partial x_j} + \nu\frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right)\right]$$

which finally leads to the Poisson's equation for pressure

$$\frac{1}{\rho}\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial u_j}{\partial x_i}\frac{\partial u_i}{\partial x_j}.$$

B.7. Relations of density, pressure and temperature and Mach number

Introducing the Mach number into the equation of the stagnation enthalpy $h_s = h + \frac{u^2}{2}$ and assuming an perfect gas ($c_P = const$, chapter A.4) one yields

$$c_pT_s = c_pT + \frac{c^2\,Ma^2}{2} = c_pT + \frac{\kappa\,R_M\,T\,Ma^2}{2}$$

which can be rewritten to

$$\frac{T_s}{T} = 1 + \frac{\kappa - 1}{2}Ma^2.$$

[107]The shear viscosity μ can be found to be approximately proportional to $\sqrt{T}$ using kinetic theory. Therefore this is equivalent to assuming a nearly constant temperature across the flow field.

Assuming isentropy (section A.6) of the flow field one yields

$$\frac{p_s}{p} = \left(\frac{T_s}{T}\right)^{\frac{\kappa}{\kappa-1}} = \left(1 + \frac{\kappa-1}{2} Ma^2\right)^{\frac{\kappa}{\kappa-1}}$$

and

$$\frac{\rho_s}{\rho} = \left(\frac{T_s}{T}\right)^{\frac{1}{\kappa-1}} = \left(1 + \frac{\kappa-1}{2} Ma^2\right)^{\frac{1}{\kappa-1}} .$$

B.7.1. Flow regimes

We can use the arguably two most important dimensionless numbers of fluid flow, the Mach and the Reynolds number to give an overview of different flow regimes as shown in figure B.1.[108]

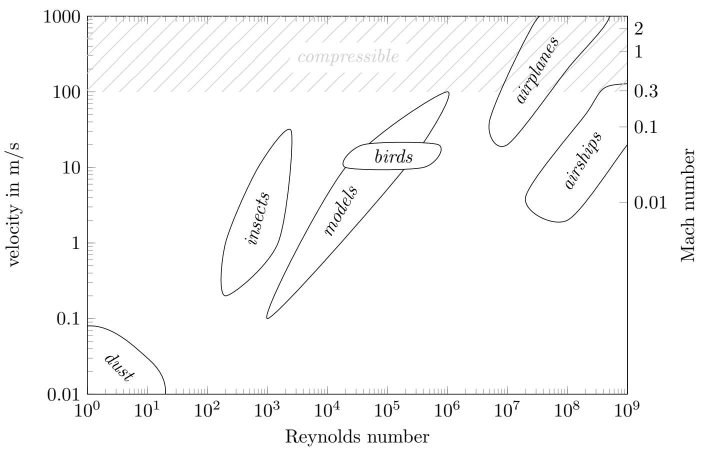


Figure B.1.: Typical dimensionless numbers for different objects moving in air [108]

Flow in most technical applications is highly turbulent with Reynolds numbers of 10^6 or higher. Only flows in porous media or flows involving particles such as dust are often strictly laminar.

B.8. Inviscid flow: Euler equations

One can prove that the work stemming from simple compression due to an external pressure is recoverable while the terms in the stress tensor proportional to the viscosity μ correspond to dissipative losses. Assuming no energy losses (adiabatic inviscid flow) one can simplify the governing conservation equations neglecting the viscous stresses as well as the heat flux leading to the so called Euler equations given by table B.2.[109]

[108]This should by no means be an accurate representation but rather illustrate the approximate magnitude and correlation of the corresponding non-dimensional numbers: The speed of sound is not constant in the atmosphere but varies with altitude and temperature and hence the two vertical axis, velocity and Mach number, are not connected through a constant factor. This variation accounts though only for roughly 10% and can be neglected on a logarithmic plot.

[109]An inviscid fluid (perfect fluid) only resists compression (normal pressure) but does not resist fluid flow.

These simplified equations may be used to describe highly turbulent flow where the inertial forces prevail over the viscous forces.

$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_j)}{\partial x_j} = 0$	*Continuity equation* conservation of mass
$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial (\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_j} + \rho g_i$	*Momentum equation* (Navier-Stokes equation) conservation of momentum
$\frac{\partial (\rho e)}{\partial t} + \frac{\partial (\rho u_j e)}{\partial x_j} = -\frac{\partial (p u_j)}{\partial x_j} + \rho u_j g_j$	*Energy equation* conservation of energy

Table B.2.: The conservation equations of fluid mechanics in differential notation for an adiabatic inviscid fluid (Euler equations)

B.9. Bernoulli equation

In the case of steady flow of a fluid with constant density, without losses due to friction, one can follow a single fluid particle along its streamline and identify three different types of energies that must be in balance: *Pressure, kinetic and potential energy*. In this case the energy along a streamline must be constant according to the so called Bernoulli equation

$$p_i + \frac{\rho \vec{u}_i^2}{2} + \rho g h_i = const.$$

When neglecting the potential energy term this results in an even simpler equation

$$p_i + \frac{\rho \vec{u}_i^2}{2} = p_i + q_i = p_s$$

where the kinetic energy term q is often also referred to as dynamic pressure. For an incompressible flow the maximum pressure is limited: If we bring the fluid to rest isentropically (adiabatic and reversible) the entire kinetic energy is converted to pressure resulting in the so called *stagnation pressure* p_s.

B.10. Fluid structure interactions

Flow past a body introduces local changes in velocity: Kinetic energy is converted to pressure and vice versa. This exerts a force on the body (figure B.2).

B.10.1. Pressure coefficient

The pressure coefficient is a dimensionless number describing the relative pressure throughout a flow field: For *every point* a pressure coefficient can be calculated using the local pressure according to

$$C_P := \frac{p - p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2}.$$

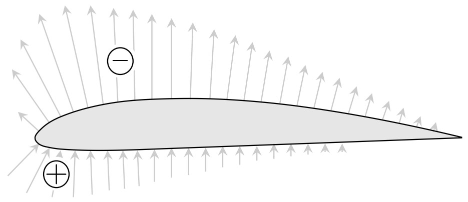


Figure B.2.: Schematic pressure distribution around an airfoil

Incompressible flow

For potential flows the Bernoulli equation (section B.9) holds and the pressure coefficient can be see as the ratio between kinetic energy in the free stream and the local dynamic pressure. Meaning the pressure coefficient in a stagnation point reaches its *maximum*, a value of *one*.

Compressible flow

In the case of compressible flow the dynamic pressure is no longer a good representation of the difference between stagnation and static pressure: The pressure coefficient might take values greater than one.

Pseudo-incompressible flow

In pseudo-incompressible solvers such as LBM in subsonic simulations one generally specifies the flow velocity at the inlet and the pressure at the outlet. The domain in between generally introduces losses in form of a pressure drop meaning that the pressure at the inlet and the velocity at the inlet have to be chosen as ρ_∞ and u_∞ respectively else the pressure coefficient might be greater than one which would be un-physical.

B.10.2. Drag coefficient

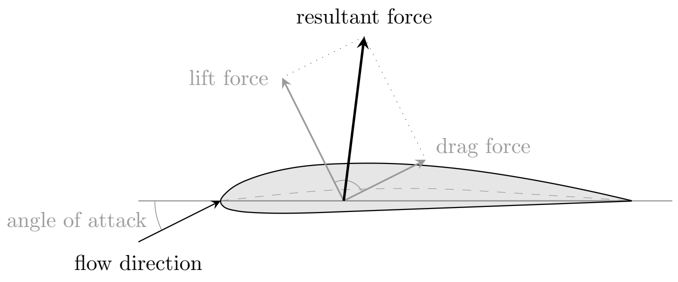


Figure B.3.: Drag and lift forces as a consequence of fluid flow around an airfoil

The force exerted by the pressure distribution around a body may be split up into two components: The component in flow direction opposing the motion is referred

to as *drag force* and can be broken down into two contributions: *Pressure drag* (form drag) depends on the shape of the object whereas *skin friction* is related to the friction between the moving object and the fluid. The other component perpendicular to the flow direction is called the *lift force* (figure B.3).
In order to quantify these two forces for different flows two additional dimensionless numbers are introduced: the drag and the lift coefficient.
The *drag coefficient* is defined by

$$C_D := \frac{2\,F_D}{\rho_\infty U_\infty^2 A_D}$$

where F_D is the drag force acting on the structure and A_D is the reference area, usually the projected area in flow direction. C_D is *not a constant* but varies depending on several parameters like the orientation of flow, the Reynolds number and in the case of compressible flow also the Mach number. A lower drag coefficient means less resistance and a $C_D = 1$ corresponds to fluid being brought to rest with a uniform stagnation pressure across the reference area.

B.10.3. Lift coefficient

Analogously to the drag coefficient

$$C_L := \frac{2\,F_L}{\rho_\infty U_\infty^2 A_L}$$

defines the lift coefficient where F_L is the lift force and A_L the reference area.

B.11. Strouhal number

Some flows exhibit *oscillating flow patterns* depending on the Reynolds number of the flow. A simple flow around a cylinder for example leads to alternating vortex shedding down-stream on top and bottom of the cylinder. This behaviour can be described by another characteristic number, the *Strouhal number*, defined as

$$St := \frac{f\,L}{U}$$

where f is the predominant vortex shedding frequency which can be determined by an FFT-analysis (appendix I.4.3.1) of an appropriate quantity like the lift coefficient.

Appendix C.

Kinetic theory of gases

Kinetic theory generally describes a *dilute gas* as a *large number of interacting particles*. The employed tools range from very simply one-dimensional models with single particles speeds to models based on distribution functions and complex far-field interactions. [1] Remarkably such simplified models already allow qualitative or even quantitative statements about processes in a real gas and *continuum* properties can be constructed as a *borderline case in thermodynamic equilibrium*. This makes kinetic theory a more general theory than continuum mechanics but yet requires simplifications or heavy numerical simulations to solve the complex equations involved.

C.1. Basic measures of kinetic theory

Instead of the density ρ for dilute gases generally the *particle density n*

$$n := \frac{N}{V},$$

where N are the number of particles in a certain volume V, is used. The corresponding density can be calculated using the mass of an individual particle m_P according to

$$\rho = n\, m_P = \frac{N}{V} m_P = \frac{m}{V}.$$

C.2. A simplified kinetic model

A very simplified kinetic gas model consisting of molecules that travel into each direction of a Cartesian coordinate system without collisions between the particles [110] can lead to viable approximate results for various physical quantities. Such a model is used in the following section to derive some basic relationships in kinetic theory.

[110]Note that such a system is theoretically instable and unphysical: Even if it was possible to create such a configuration it would homogenise due to collisions after a short time.

C.2.1. Pressure

Taking into consideration a cuboid container with the side length L we can derive the pressure from the change of momentum as follows. The collisions of particles with the walls of the container are assumed perpendicularly and elastic. The resulting change of momentum for a single particle is given in this case by

$$\Delta p = p_{in} - p_{out} = 2p_{in} = 2m_P v_x.$$

The particle will continue moving and collide with the opposite wall and hit the first wall again after

$$\Delta t = \frac{2L}{v_x}.$$

The force due to this particular particle can be calculated to

$$F_P = \frac{\Delta p}{\Delta t} = \frac{m_P v_x^2}{L}$$

and the total force of N particles on the wall can be obtained by introducing a mean velocity $\overline{v_x}$

$$F = \frac{\Delta p}{\Delta t} = \frac{N m_P \overline{v_x}^2}{L}.$$

Now since the motion is assumed randomly the mean square velocities in all three spatial directions have to be equal given a large number of particles: $\overline{v_x}^2 = \overline{v_y}^2 = \overline{v_z}^2$ and hence

$$\overline{v}^2 = \overline{v_x}^2 + \overline{v_y}^2 + \overline{v_z}^2 \approx 3\overline{v_x}^2$$

Therefore the pressure can be written as force by area

$$p = \frac{F}{A} = \frac{N m_P \overline{v}^2}{3L^3} = \frac{N m_P \overline{v}^2}{3V} = \frac{\rho \overline{v}^2}{3} = \frac{2}{3} n \frac{m_P}{2} \overline{v}^2.$$

C.2.2. Thermal energy

The thermal energy does not explicitly appear in the kinetic theory. Instead it emerges from the movement of particles, meaning the mean kinetic energy of a mono-atomic gas[111] must be equivalent to the thermal energy[112]

$$E_{kin} = \frac{m_P \overline{v}^2}{2} = \frac{3}{2} k_B T = E_{therm}.$$

And therefore the mean velocity can be calculated to

$$\overline{v} = \sqrt{3 \frac{k_B}{m_p} T} = \sqrt{3 R_m T}.$$

[111]Monoatmic gases have only three degrees of freedom resulting from the linear motion while gases made of more complex molecules have additional 'internal' degrees of freedoms such as rotation and vibration.

[112]This energy is distributed equally among the three degrees of freedom.

C.3. Particle distributions

More complex models in kinetic theory involve particle distributions that describe the probability of finding a certain gas molecule in space with a certain velocity. The following section is supposed to extend the basics elaborated in the core of the thesis.

C.3.1. Equilibrium

A thermodynamical system can either be in equilibrium or non-equilibrium. In an equilibrium state no change is visible on a macroscopic level which can happen in two different cases, global and local equilibrium. In a *global equilibrium* the system is uniform throughout the whole domain (no gradients). Even if boundary conditions might prevent a system from returning to a global equilibrium, the relevant thermodynamic variables might only changing slowly in space and time and thus any small macroscopic region can be described accurately by an equilibrium distribution in some neighbourhood. This condition is referred to as *local equilibrium*.
Any other situation where either of the above conditions are not met is called non-equilibrium. The special case of a time independent non-equilibrium system is referred to as *steady state*. [93], [94]
If a system in non-equilibrium is isolated from its environment and every cause of inhomogeneity (like heat sources) is eliminated it will first return to a local and then in the long run to global equilibrium.[113]

C.3.2. Derivation of the Maxwell-Boltzmann equilibrium distribution

Just arguing with *symmetries and conservation of moments* one is able to predict a possible distribution function in a state of *equilibrium*, the so called Maxwell-Boltzmann distribution for mono-atomic gases.
In a system without boundary conditions with a certain initial perturbation it seems reasonable to assume that after some time the perturbation will even out across the domain and the system will reach an equilibrium distribution that is isotropic in velocity space. This means it should be possible to write an equilibrium distribution for velocity as the product of three one-dimensional distribution functions[114] in terms of the deviation of the microscopic velocities from the macroscopic velocity $\vec{v} = \vec{\xi} - \vec{u}$.[115]

$$f^{(eq)}(\vec{v}^2) = f^{(eq)}(v_x^2 + v_y^2 + v_z^2) = f_{1D}^{(eq)}(v_x^2)\, f_{1D}^{(eq)}(v_y^2)\, f_{1D}^{(eq)}(v_z^2)$$

If the magnitude of velocity $\vec{v}^2$ is held constant also $f^{(eq)}(\vec{v}^2)$ must be a constant and therefore

$$ln(f^{(eq)}(\vec{v}^2)) = ln(f_{1D}^{(eq)}(v_x^2)) + ln(f_{1D}^{(eq)}(v_y^2)) + ln(f_{1D}^{(eq)}(v_z^2)) = const$$

[113]Hence one might think of a non-equilibrium system near equilibrium as a small perturbation of the equilibrium system and describe the system mathematically using the later discussed perturbation theory.
[114]This corresponds to a joint probability.
[115]Deviation from the mean value

holds. The easiest non-trivial distribution function fulfilling this criterion is given by $f_{1D}^{(eq)}(v_x^2) = A - Bv_x^2$ due to

$$ln(f_{1D}^{(eq)}(v_x^2)) + ln(f_{1D}^{(eq)}(v_y^2)) + ln(f_{1D}^{(eq)}(v_y^2)) = 3A + B(v_x^2 + v_y^2 + v_z^2) = 3A - B|\vec{v}|^2.$$

Therefore one possible equilibrium distribution function takes the form of

$$f^{(eq)}(|\vec{v}|) = e^{3A}e^{-B|\vec{v}|^2} = Ce^{-B|\vec{v}|^2}.$$

The zeroth moment of any particle distribution must be equal to the particle density n.[116] The resulting integral converges only for negative exponents $-B < 0$. Again we assume that the integral can be split up into one-dimensional solutions (appendix I.3.3) according to

$$\iiint \left(\frac{B}{\pi}\right)^{\frac{3}{2}} e^{-B\vec{x}^2} d\vec{x} = \left(\int \sqrt{\frac{B}{\pi}} e^{-Bx^2} dx\right) \left(\int \sqrt{\frac{B}{\pi}} e^{-By^2} dy\right) \left(\int \sqrt{\frac{B}{\pi}} e^{-Bz^2} dz\right)$$

Integration (section I.3.3) and enforcing the zero-th momentum leads to

$$C = n\left(\frac{B}{\pi}\right)^{\frac{3}{2}}$$

meaning this equilibrium distribution must take the form

$$f^{(eq)}(|\vec{v}|) = n\left(\frac{B}{\pi}\right)^{\frac{3}{2}} e^{-B|\vec{v}|^2}.$$

Enforcing the second order momentum

$$\rho e = \frac{n\,k_B\,T}{2} = \frac{m_P}{2} \iiint \vec{v}^2 f^{(eq)}(|\vec{v}|).$$

yields[117]

$$B = \frac{m_P}{2\,n\,k_B\,T} = \frac{1}{2\,R_m\,T}.$$

This means the so called Maxwell-Boltzmann distribution in three-dimensional space takes the form of a Gaussian distribution (appendix I.2.1)

$$f^{(eq)}(\vec{x}, |\vec{v}|, t) = n\frac{1}{(2\pi R_m T)^{\frac{3}{2}}} e^{-\frac{|\vec{v}|^2}{2R_m T}}.$$

C.3.3. Moments of particle distributions

The moments of the particle distribution can be used to calculate the corresponding macroscopic properties in continuum mechanics. The following moments are integrated over the absolute velocities of the particles but would take the same form for the relative velocity $\vec{v} = \vec{\xi} - \vec{u}$ due to $d\vec{v} = d\vec{\xi}$.

[116] $N = \iint f d\vec{x} d\vec{\xi}$ leads to $n = \lim_{\Delta V \to 0} = \frac{\Delta N}{\Delta V} \int f d\xi$ if f can be assumed as spatially constant

[117] $\frac{k_B}{m_P} = \frac{R}{M} = R_m$

Moment	*Macroscopic property*
$m_P \int f d\vec{\xi} = n\, m_P = \rho$	Density
$m_P \int (\xi_i - u_i) f d\vec{\xi} = 0$	Mean molecular velocity
$m_P \int \xi_i f d\vec{\xi} = \rho u_i$	Momentum per volume
$\frac{m_P}{2} \int \left\|\vec{\xi} - \vec{u})\right\|^2 f d\vec{\xi} = \rho e = \rho \frac{3}{2} RT$	Internal energy
$m_P \int \left\|\vec{\xi}\right\|^2 f d\vec{\xi} = \rho(e + \frac{\|\vec{v}\|^2}{2})$	Total energy
$m_P \int (\xi_i - u_i)(\xi_j - u_j) f d\vec{\xi} = \sigma_{ij}$	Stress tensor
$\frac{m}{3} \int \left\|\vec{\xi} - \vec{u}\right\|^2 f d\vec{\xi} = -p$	Pressure
$m_P \int \xi_i \xi_j f d\vec{\xi} = \rho u_i u_j - p\delta_{ij} + \tau_{ij}$	Momentum flux and stress tensor
$\frac{m_P}{2} \int \left\|\vec{\xi} - \vec{u}\right\|^2 (\xi_i - u_i) f d\vec{\xi} = q_i$	Heat flux
$\frac{m_P}{2} \int \left\|\vec{\xi}\right\|^2 \xi_i f d\vec{\xi} = \rho u_i (e + \frac{\|\vec{u}\|^2}{2}) + v_j p_{ij} - q_i$	Energy flux

Table C.1.: Relation between the microscopic moments and the macroscopic properties of a mono-atomic ideal gas[1]

C.3.4. Collision integral: Stoßzahlansatz

Boltzmann introduced a general collision operator for collisions of two particles, assuming collisions of multiple particles are very rare. A fixed volume in phase space might lose and gain particles from adjacent domains due to collisions. This gain and loss can be expressed using an equivalent collision area A_c and the relative velocity $g = \left|\vec{\xi}_1 - \vec{\xi}\right|$ as well as assuming the reversibility of single collisions to

$$\Omega = \int\limits_{A_c} \int\limits_{\vec{\xi}_1} \left(f' f_1' - f f_1\right) g d\vec{\xi}_1 dA_c$$

where indexes with apostrophe reflect the scattered quantities.[118]

C.3.5. Dimensionless Boltzmann equation

One can also introduce dimensionless variables into the Boltzmann equation

$$f^* = \frac{f\,(3RT)^{\frac{3}{2}}}{n} \qquad x_i^* = \frac{x_i}{L} \qquad \xi_i^* = \frac{\xi_i}{\sqrt{3RT}}$$

$$t^* = \frac{t\,\sqrt{3RT}}{L} \qquad g^* = \frac{g}{\sqrt{3RT}} \qquad dA_c = A_c^*\, n\, \lambda$$

[118]Note: The multiplication of the two distribution functions in the collision operator corresponds to the joint distribution of two uncorrelated distributions. This reflects the molecular chaos hypothesis.

finding the Knudsen *Kn* number as dimensionless parameter

$$\frac{\partial f^*}{\partial t^*} + \xi_i^* \frac{\partial f^*}{\partial x_i^*} = \frac{1}{Kn} \int\limits_{A_c^*} \int\limits_{\vec{\xi}_1^*} \left(f^{*\prime} f_1^{*\prime} - f^* f_1^* \right) g^* d\vec{\xi}_1^* dA_c^*.$$

In the continuum limit in thermodynamic equilibrium, for Knudsen numbers tending to zero, the collision term becomes dominant. This is equal to a vanishing integrand

$$f^{*\prime} f_1^{*\prime} - f^* f_1^* = 0.$$

The *number of particles entering the domain must be equal to the number of particles leaving* it. The *Maxwell-Boltzmann distribution exactly fulfils* this condition: As the system is ruled by the $\mathcal{H}$-theorem this means that all non-equilibrium distributions tend over time to a Maxwell-Boltzmann distribution.

Appendix D.

Multi-component flow

D.1. Basic definitions for mixtures

A *species* is an ensemble of chemically identical molecular entities that can explore the same set of molecular energy levels on the time scale of the experiment whereas a particular *phase* is a chemically and physically uniform quantity of matter that can be separated mechanically and may consist of a single substance or of different substances. A system constituted by either different species and/or different phases is termed a multi-species or multi-phase *mixture*.[95]

D.1.1. Scale of separation and homogeneous mixtures

One might use the interface between the generic phase and the other phases to define a *characteristic scale of separation*. In *disperse flow* this length scale will be significantly smaller than the characteristic length scale of the problem L while for *separated flows* it will be of the same order or larger leading to completed separated or film flow.
If the characteristic scale of separation is even smaller than the smallest length scale modelled in the chosen description (such as the lattice size) each control volume contains representative samples of each phase and the mixture is termed *homogeneous mixture*. The following description is based on such a homogeneous mixture.[95]
Generally one species' concentration in a homogeneous mixture is predominant with the other components and is therefore referred to as *solvent* whereas the other components are comparably small and are therefore referred to as *dilute*.

D.2. Mole, mass and volume fractions

For describing multi-component mixtures one may express the *ratio of constituents* in different ways. Most commonly either *mole* or *mass fractions* are deployed. The corresponding *sum* of all fractions obviously must be equal to *unity*. Molar averaging is mainly used for chemistry whereas mass averages are generally used for liquids.

D.2.1. Mole fractions

In chemistry generally the *mole fractions* defined by the ratio of amount of an individual constituent n_i to the amount of all constituents n

$$\chi_i := \frac{n_i}{n}$$

is used.

D.2.2. Mass fractions

More commonly in CFD simulations the *mass fractions* are adopted instead. The relation to the corresponding mole fractions can be established through the molar mass of the individual component M_i and the molar mass of the mixture M according to

$$Y_i := \frac{m_i}{m} = \chi_i \frac{M_i}{M}.$$

D.2.3. Molar concentration, partial volume and volume fractions

The *molar concentration* of an individual component c_i is defined as the amount of constituent n_i per volume of the mixture V and can be linked through the mole fractions to the molar concentration of the mixture c by

$$c_i := \frac{n_i}{V} = \chi_i c.$$

The ratio between the volume of a single component V_i[119] to the total volume of the mixture V is termed *volume fraction* ϕ_i

$$\phi_i := \frac{V_i}{V} = c_i V_i.$$

For an ideal gas the volume fractions can be found to be equal to the mole fractions due to the equation of state

$$\phi_i = \frac{V_i}{V} = \frac{n_i}{n} = \chi_i.$$

D.2.4. Mixture averages

This leaves us with several options of obtaining mixture averages. For some quantities it is obvious which way of averaging must be used, while in other cases different possibilities may be found in literature which might be confusing at first. The following section should help clarify this.

D.2.4.1. Molar mass

The *molar mass* obviously is averaged using the mole fractions χ_i according to

$$M = \sum_i \chi_i M_i.$$

[119]Partial volume in the case of an ideal gas

D.2.4.2. Mixture density

The *density* of the component i can be calculated to

$$\rho_i = \frac{m_i}{V}$$

and therefore the mixture density ρ is given by

$$\rho = \frac{m}{V} = \sum_i \frac{m_i}{V} = \sum_i \rho_i.$$

D.2.4.3. Molar-, mass- and volume-averaged velocity

It is not obvious which average should be used for the velocity and therefore several annotations exist in the literature. One common way of calculating the mixture average velocity is given by the *mass-averaged* (barycentric) velocity

$$\vec{u}^{\mathrm{Y}} := \sum_i \mathrm{Y}_i \vec{u}_i$$

another by the *molar-averaged* velocity

$$\vec{u}^{\chi} := \sum_i \chi_i \vec{u}_i$$

and sometimes the *volume-averaged velocity*

$$\vec{u}^{\Phi} := \sum_i \phi_i \vec{u}_i$$

is taken.

D.3. Total, convective and diffusive flux

One thinks of the overall (total) flux N as a *superimposition of a convective and a diffusive flux J*. The fluxes might be denoted as *mass* (lower case letters) or *molar fluxes* (capital letters) with their corresponding mass and molar averaged velocities.
In the case of mass fluxes this leads to

$$\vec{n}_i = \mathrm{Y}_i \rho \vec{u}_i = \rho_i \vec{u}_i = \vec{j}_i + \rho \vec{u}^{\mathrm{Y}}$$

with the total mass flux

$$\vec{n} = \sum_i \vec{n}_i = \sum_i \rho_i \vec{u}_i = \rho \vec{u}^{\mathrm{Y}}$$

and similarly for molar fluxes

$$\vec{N}_i = \chi_i c \vec{u}_i = c_i \vec{u}_i = \vec{J}_i + c_i \vec{u}^{\Phi}$$

with the total molar flux

$$\vec{N} = \sum_i \vec{N}_i = \sum_i c_i \vec{u}_i = c \vec{u}^{\Phi}.$$

D.4. Fick's laws

D.4.1. Derivation of Fick's first law

In a one dimensional steady-state diffusion process, where particles at one point diffuse equally into both directions, the number of particles moving in the positive x direction for a discrete system is given by

$$-\frac{1}{2}\left(n_i(x+\Delta x,t)-n_i(x,t)\right)$$

and the corresponding flux per area element A and time step Δt is given by

$$J_i=-\frac{1}{2\,A\,\Delta t}\left(n_i(x+\Delta x,t)-n_i(x,t)\right)$$

This can be rewritten to

$$J_i=-\frac{\Delta x^2}{2\,\Delta t}\left(\frac{n_i(x+\Delta x,t)-n_i(x,t)}{A\,\Delta x^2}\right)$$

and introducing the molar concentration c_i as well as the diffusion constant D_i

$$D_i=\frac{\Delta x^2}{2\Delta t}$$

this yields

$$J_i=-D_i\left(\frac{c_i(x+\Delta x,t)-c_i(x,t)}{\Delta x}\right)$$

which for the case of $\Delta x\rightarrow 0$ leads to

$$J_i=-D_i\frac{\partial c_i}{\partial x}.$$

For a three dimensional system the gradient replaces the partial derivative

$$\vec{J}_i=-D_i\nabla c_i=-c\,D_i\nabla\chi_i.$$

D.4.1.1. Alternative notations

Fick's law might also be denoted in terms of the mass fractions Y

$$\vec{j}_i=-\rho D_i\nabla \mathrm{Y}_i.$$

D.4.2. Derivation of Fick's second law

Assuming again a one-dimensional system where now concentration changes over time as well as due to diffusion

$$\frac{\partial c_i}{\partial t}+\frac{\partial J_i}{\partial x}=0$$

one yields with Fick's first law

$$\frac{\partial c_i}{\partial t} - \frac{\partial}{\partial x}\left(D_i \frac{\partial c_i}{\partial x}\right) = 0$$

which assuming a constant diffusion coefficient D finally yields *Fick's second law*

$$\frac{\partial c_i}{\partial t} = D_i \nabla^2 c_i.$$

D.4.3. Correction velocity

If the Fick's law is modified to account for multi-component flow it does not guarantee mass conservation. This property can be improved by introducing an additional *correction velocity* as

$$\vec{u}_c = -\sum_i Y_i \vec{u}_i$$

leading to the effective diffusion term

$$Y_i \vec{u}_i^* = Y_i \vec{u}_i + Y_i \vec{u}_c = -D_i \nabla Y_i + \sum_j D_j \nabla Y_j.$$

D.5. Lewis number

For thermal multi-component system another dimensionless number, the *Lewis number*

$$Le := \frac{a}{D_i} \qquad \frac{\text{thermal diffusion}}{\text{molecular diffusion}}$$

which is related to the Schmidt and Prandtl numbers by

$$Le = \frac{Sc}{Pr}$$

can be found in the literature.

D.6. Non-binary multi-component flow

When dealing with true non-binary multi-component flow the Fick model loses its validity but my still be used with sufficient accuracy leading to a non-symmetric collision matrix. Other common approaches such as the *effective gas diffusion model* try to reduce multi-component flow to a binary mixture introducing a composite gas. More advanced diffusion models such as the *Maxwell-Stefan model* (appendix D.7) unlike Fick's laws are still adequate for flow of multiple components but fail in porous media where the particles collide often with the walls. In this case so the so called *dusty gas model* may still be used. [96]
For dilute systems, which is the rule rather than the exception, the multi-component effects are though minor. Often the Fick's equation is generalised using non-symmetric multi-component diffusion coefficients D_{ij}. The resulting equation can be rewritten in a matrix form where the diagonal (main) terms are similar to the binary values and the diagonal (cross) terms are significantly smaller (10% or less of the main terms).

D.7. Maxwell-Stefan system

A more general model than the Fick's law is provided by a special species momentum conservation equation derived on the basis of a dilute gas model, the so called *Maxwell-Stefan* system

$$\nabla\chi_i = \sum_j \frac{\chi_i\chi_j}{Đ_{ij}}(\vec{u}_j - \vec{u}_i) + (Y_i - \chi_i)\frac{\nabla p}{p} + \frac{\rho}{p}\sum_j Y_i Y_j(\vec{g}_i - \vec{g}_j) + \sum_j \frac{\chi_i\chi_j}{Đ_{ij}}\left(\frac{Đ_j^T}{Y_j} - \frac{Đ_i^T}{Y_i}\right)$$

where χ_i is the *mole fraction* of a particular component and $Đ$ are the corresponding *diffusion coefficients*.[120]
There are four different diffusion mechanism: diffusion in between species, diffusion due to pressure, body forces and thermal diffusion (thermophoresis often also termed Soret-effect). Neglecting all the secondary diffusion effects leads to

$$\nabla\chi_i = \sum_j \frac{\chi_i\chi_j}{Đ_{ij}}(\vec{u}_j - \vec{u}_i).$$

As can be seen this approach avoids average velocities although somewhat obscuring the insight. It is very challenging to determine the particular diffusion coefficients especially for mixtures of multiple components which in general are different from those in Fick's law. The equation can be rewritten to a matrix form and be solved analytically where the main advantage over Fick's law is the symmetry of the coefficients involved $Đ_{ij} = Đ_{ji}$. A difficulty arises though from coupling the Maxwell-Stefan system with the mass fractions. This is probably the reason this model is not yet extensively used.

D.7.1. Derivation of the Maxwell-Stefan system for binary flow

For the derivation we assume the collision of two elastic particles that conserves momentum

$$m_1(\vec{u}_1 - \vec{u}_1') + m_2(\vec{u}_2 - \vec{u}_2') = 0$$

as well as energy

$$\frac{m_1}{2}(\vec{u}_1^2 - \vec{u}_1'^2) + \frac{m_2}{2}(\vec{u}_2^2 - \vec{u}_2'^2) = 0.$$

The momentum exchanged through the collision can then be calculated to

$$m_1(\vec{u}_1 - \vec{u}_1') = \frac{2m_1m_2(\vec{u}_1 - \vec{u}_2)}{m_1 + m_2}.$$

This means some sort of proportionality of the form

$$\chi_1 \propto \chi_1\chi_2(\vec{u}_1 - \vec{u}_2)$$

must hold. Integrating the above equation over a particular surface S and applying Gauss's divergence theorem leads to

$$-\int_S \chi_1\vec{n}dS = -\int_V \nabla\chi_1 dV.$$

[120]Due to momentum conservation the binary diffusion coefficients have the symmetry $Đ_{ij} = Đ_{ji}$ but depend themselves on temperature, pressure and the characteristics of the two species involved.

Introducing the binary diffusion coefficient $Đ_{12}$ as a proportionality constant this yields the Maxwell-Stefan law for binary mixtures

$$\nabla \chi_1 = -\frac{\chi_1 \chi_2}{Đ_{12}} (\vec{u}_2 - \vec{u}_1).$$

D.7.2. Derivation of Fick's first law from the Maxwell-Stefan system

For binary flow the Maxwell-Stefan system can be shown to be equivalent to Fick's first law due to

$$\vec{u}^V = c_1 V_1 \vec{u}_1 + c_2 V_2 \vec{u}_2 = V_1 \vec{N}_1 + V_2 \vec{N}_2$$

which can be written as

$$\vec{N}_2 = \frac{\vec{u}^V - V_1 \vec{N}_1}{V_2}$$

Inserting this into the Maxwell-Stefan equation leads to

$$\nabla \chi_1 = -\frac{\chi_1 \chi_2}{Đ_{12}} (\vec{u}_2 - \vec{u}_1) = -\frac{\chi_2 \vec{N}_1 - \chi_1 \vec{N}_2}{c\, Đ_{12}} = -\frac{\vec{N}_1 - c_1 \vec{u}^V}{Đ_{12}\, c^2\, V_2} = -\frac{\vec{J}_1}{Đ_{12}\, c^2\, V_2}.$$

and further to

$$\vec{J}_1 = -Đ_{12}\, c^2\, V_2\, \nabla \chi_1.$$

For an ideal gas the partial molar volume is equal to the reciprocal of the total molar concentration and thus $c\, V_2$ equals unity. This results in Fick's first law

$$\vec{J}_1 = -Đ_{12}\, c\, \nabla \chi_1.$$

The two diffusion coefficients, the Fickian diffusivity D_1 and the Maxwell-Stefan coefficients $Đ_{12}$ are identical for the special case of binary ideal gas flow.

Appendix E.

Lattice-Boltzmann

The following appendix contains various extensive derivations in the context of the Lattice-Boltzmann methods that were omitted in thesis due to their complexity and extent.

E.1. Derivation of Lattice-Boltzmann

There are multiple ways of deriving the macroscopic equations of conservation from the mesoscopic Boltzmann equation through discretisation and asymptotic analysis. The one presented in this chapter takes the form of a multiple scale perturbation analysis, namely the Chapman-Enskog expansion.
The following derivation using the common $D2Q9$ lattice and the basic *BGK collision operator* is based on [97]–[100] with the addition of multiple intermediate steps and is supported by a chapter covering the maths behind it in detail, such as quadratures and perturbation theory (appendix J).

E.1.1. Discrete Lattice-Boltzmann

In the second chapter it was shown that the continuous Boltzmann equation preserves the familiar conservation equations but the system of equations is not yet closed. Assuming that the distributions are nothing but a perturbation of the equilibrium distribution one is able to retrieve the Navier-Stokes equations. In order to be used as a numerical method for simulating fluid flow, the Boltzmann equation has to be *discretised* in time as well as in space and then the same procedure has to be applied. Generally *cubic meshes*, classified by the $DnQm$ notation where n is the number of dimension of a lattice and m are the symmetric velocities of a single lattice, are chosen. In this section it will be shown how a suiting discrete model can be derived using the widely used $D2Q9$ model as an example. For the sake of simplicity and clearness but without losing generality we will stick to the Bhatnagar-Gross-Krook (BGK) collision operator. More complicated models like the $D3Q19$ or the $D3Q27$ with various collision operators can be derived in a similar fashion.

E.1.1.1. Discretisation in time

The continuous Boltzmann equation can be written as

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \nabla f + \frac{\vec{F}}{\rho} \cdot \nabla_{\vec{\xi}} f = \frac{1}{\lambda}(f^{(eq)} - f).$$

We *neglect the force term* for the complete derivation but it may be added afterwards independently with a more basic approach that does not require the entire derivation to be adjusted (section E.2). This leaves us with the very basic equation

$$\frac{\partial f}{\partial t} + \vec{\xi} \cdot \nabla f = \frac{1}{\lambda}(f^{(eq)} - f)$$

that, combining the remaining terms on the left hand side to the material derivative (appendix I.3.1), may be rewritten to

$$\frac{Df}{Dt} = \frac{1}{\lambda}(f^{(eq)} - f).$$

Bringing the terms including f on the left hand side and using a trick by multiplying the equation with $e^{\frac{t}{\lambda}}$ it is possible to combine the left hand side to a single term

$$\frac{D}{Dt} e^{\frac{t}{\lambda}} f = e^{\frac{t}{\lambda}} \frac{Df}{Dt} + e^{\frac{t}{\lambda}} \frac{1}{\lambda} f = e^{\frac{t}{\lambda}} \frac{1}{\lambda} f^{(eq)}$$

By *integrating this equation over a time interval* Δt we get:

$$\left[e^{\frac{t}{\lambda}} f \right]_0^{\Delta t} = e^{\frac{\Delta t}{\lambda}} f(\vec{x} + \Delta t \, \vec{\xi}, \vec{\xi}, t + \Delta t) - f(\vec{x}, \vec{\xi}, t) = \frac{1}{\lambda} \int_0^{\Delta t} e^{\frac{t'}{\lambda}} f^{(eq)}(\vec{x} + t' \, \vec{\xi}, \vec{\xi}, t + t') dt'.$$

Now we assume that the function $f^{(eq)}$ is constant during the time step Δt as it will be comparably small and hence the right hand side of the equation above simplifies to

$$f^{(eq)} \frac{1}{\lambda} \int_0^{\Delta t} e^{\frac{t'}{\lambda}} dt' = f^{(eq)} \left[e^{\frac{t}{\lambda}} \right]_0^{\Delta t} = f^{(eq)} \left[e^{\frac{\Delta t}{\lambda}} - 1 \right]$$

This leads to

$$f(\vec{x} + \Delta t \, \vec{\xi}, \vec{\xi}, t + \Delta t) = f^{(eq)} \left[1 - e^{-\frac{\Delta t}{\lambda}} \right] + e^{-\frac{\Delta t}{\lambda}} f(\vec{x}, \vec{\xi}, t)$$

By subtracting $f(\vec{x}, \vec{\xi}, t)$ on both sides of the equation and introducing a new parameter, the *relaxation time*, $\tau = 1/\left(1 - e^{-\frac{\Delta t}{\lambda}}\right)$ we yield

$$f(\vec{x} + \Delta t \, \vec{\xi}, \vec{\xi}, t + \Delta t) - f(\vec{x}, \vec{\xi}, t) = \frac{1}{\tau} \left[f^{(eq)}(\vec{x}, \vec{\xi}, t) - f(\vec{x}, \vec{\xi}, t) \right]$$

If τ is expanded in a Laurent series (appendix I.4.2) neglecting the terms $\mathcal{O}(\Delta t^2)$ or smaller it can be see that $\frac{1}{\tau} \approx \frac{\Delta t}{\lambda}$ which is often mentioned in the literature.
This so-called Lattice-Boltzmann equation can be seen as a first order discretisation of the Boltzmann equation but through re-parametrisation it can be shown to be equivalent to the second order discretisation.[5]

E.1.1.2. Low Mach number expansion

Now we expand the equilibrium distribution function in a *small Mach number expansion* assuming that the fluid of interest is approximately *incompressible*: The density fluctuations $\Delta\rho$ are assumed to be of order $\mathcal{O}(Ma^2)$ or smaller. Therefore we decompose the Maxwell-Boltzmann distribution, which in the case of LBM is formulated in terms of the density ρ instead of the particle number n like in kinetic theory (chapter C.1)[121],

$$f^{(eq)} = \frac{\rho}{(2\pi R_m T)^{\frac{D}{2}}} e^{-\frac{(\vec{\xi}-\vec{u})^2}{2R_m T}} = \frac{\rho}{(2\pi R_m T)^{\frac{D}{2}}} e^{-\frac{\vec{\xi}}{2R_m T}} e^{-\frac{\vec{u}^2-2\vec{u}\vec{\xi}}{2R_m T}}$$

and then expand it using a second order Taylor series (section I.4.1) $Ae^{-Bx^2+2Cx} = A(1+2Cx+x^2(2C^2-B)+\ldots)$

$$f^{(eq)} = \frac{\rho}{(2\pi R_m T)^{\frac{D}{2}}} e^{-\frac{\vec{\xi}^2}{2R_m T}} \left[1 + \frac{\vec{\xi}\vec{u}}{R_m T} + \frac{(\vec{\xi}\vec{u})^2}{2(R_m T)^2} - \frac{\vec{u}^2}{2R_m T} + \cdots\right].$$

This equation is still continuous in space but we will try to find a discrete version of it that preserves the required momenta.

E.1.1.3. Discretisation of momentum space

In order to prove that on macroscopical level the discrete Boltzmann equation behaves like the Navier Stokes equations the hydrodynamic moments have to be evaluated. While in order to derive the *isothermal* Navier-Stokes equation only the *first four moments* are necessary, for thermal LBM the first five moments are required.[122] Taking into consideration higher moments one is able to derive even more complex models like the Burnett-equations. [101] The hydrodynamic moments take the following form

$$\int \vec{\xi}^m f^{(eq)} d\vec{\xi}.$$

The resulting integrals can be approximated through *multidimensional quadrature rules*, weighted sums of function values (appendix I.3.5). The velocity space has to be discretised with velocities pointing into certain directions ξ_α with their corresponding weights W_α. The form of the resulting integrals involving $e^{-x^2}f(x)dx$ due to the term in the pseudo-incompressible equilibrium is typically approximated using a Gauss-Hermite quadrature (appendix I.3.5.1). Common quadratures have though too little velocities to also preserve the energy equation, meaning that the model basically reduces to an athermal model, temperature bears no meaning. [102]
The moments contain integrals of following form

$$\int e^{-\frac{\vec{\xi}^2}{2R_m T}} \psi(\vec{\xi}) d\vec{\xi} = \sum_\alpha W_\alpha \psi(\vec{\xi}_\alpha)$$

where ψ is a polynomial of degree m and

$$f_\alpha^{(eq)}(\vec{x},t) = W_\alpha f^{(eq)}(\vec{x},\vec{\xi},t).$$

[121]One might think of this as a fictitious model gas with a particle mass of $m_P = 1$.

[122]This requires higher order tensors to be preserved and therefore also lattices with more discrete velocities.

is the discrete equilibrium distribution. From now on we assume a two dimensional $D2Q9$ model and hence with a two-dimensional quadrature $\psi(\vec{\xi}) = \xi_x^m \xi_y^n$ the integral above can be decomposed into the integration of two integrals I_m and I_n[123] where the leading term $(\sqrt{2R_mT})^{(m+n+2)}$ emerges from the substitution of the variables ξ_i by ζ_i:

$$I = \int e^{-\frac{\xi_x^2+\xi_y^2}{2R_mT}} \xi_x^m \xi_y^n d\vec{\xi} = (\sqrt{2R_mT})^{(m+n+2)} I_m I_n$$

with

$$I_i = \int_{-\infty}^{\infty} e^{-\zeta^2} \zeta^i d\zeta$$

where $\zeta = \xi_x / \sqrt{2R_mT}$ or $\xi_y / \sqrt{2R_mT}$.
For a 9 velocity discretisation we used 3 integration points in each direction using the third-order Hermite formula (appendix I.3.5.1)

$$I_i = \sum_{j=1}^{3} \omega_j \zeta_j^i$$

with the corresponding abscesses ζ_j and weights ω_j:

$$\zeta_1 = -\sqrt{\frac{3}{2}} \qquad \zeta_2 = 0 \qquad \zeta_3 = \sqrt{\frac{3}{2}}$$

$$\omega_1 = \frac{\sqrt{\pi}}{6} \qquad \omega_2 = \frac{2\sqrt{\pi}}{3} \qquad \omega_3 = \frac{\sqrt{\pi}}{6}$$

The complete integral I can be calculated using all resulting nine (three for every direction) quadrature points (using the symmetry of the weights ω_1 and ω_3):

$$I = 2R_mT\Big[\omega_2^2 \psi(\vec{0}) + \sum_{\alpha=1}^{4} \omega_1\omega_2 \psi(\vec{\xi}_\alpha) + \sum_{\alpha=5}^{8} \omega_1^2 \psi(\vec{\xi}_\alpha)\Big]$$

where the corresponding $\vec{\xi}_\alpha$-values can be found due to variable transformation of the quadrature points (hence the pre-factor $c = \sqrt{3R_mT}$). In order to emphasise their discrete nature with a meaning similar to a unit vector and to differentiate them from the microscopic velocities $\vec{\xi}$ we denote them with $\vec{e}$ instead of $\vec{\xi}$.

$$\begin{array}{llll} \vec{e}_0 = (0,0), & & & \\ \vec{e}_1 = c(1,0), & \vec{e}_2 = c(0,1), & \vec{e}_3 = c(-1,0), & \vec{e}_4 = c(0,-1), \\ \vec{e}_5 = c(1,1), & \vec{e}_6 = c(-1,1), & \vec{e}_7 = c(-1,-1), & \vec{e}_8 = c(1,-1) \end{array}$$

E.1.1.4. Discretisation of configuration space

The configuration space is discretised accordingly, meaning

$$c = \frac{\Delta x}{\Delta t} = \frac{\Delta y}{\Delta t} = \sqrt{3R_mT}$$

[123]Note for other discretisations that can not be obtained by Gauss's product rules other approaches might be needed.

has to be fulfilled.[124]
As the temperature in our isothermal equation of state

$$p = \rho R_m T,$$

bears no meaning any more we choose the grid spacing Δx as the fundamental variable instead. The *lattice speed of sound* which links pressure and denisity can be found to

$$c_s^2 = \left(\frac{\partial p}{\partial \rho}\right)_s = R_m\, T = \frac{c^2}{3}$$

and the equation of state can be rewritten to

$$p = c_s^2 \rho.$$

We combine the rather clumsy leading part of the discrete pseudo-incompressible Lattice-Boltzmann equilibrium distribution and the weights W_α to new weights

$$w_\alpha = \frac{W_\alpha}{2\pi R_m T} e^{-\frac{\vec{\xi}^2}{2R_m T}}$$

that can be found by comparing the two equations to

$$w_\alpha = \begin{cases} 4/9, & \alpha = 0 \\ 1/9, & \alpha = 1,2,3,4 \\ 1/36, & \alpha = 5,6,7,8 \end{cases}$$

The *equilibrium distribution function* can be rewritten, using the new weights as well as the lattice speed of sound, to

$$f_\alpha^{(eq)}(\vec{x}, \vec{e}_\alpha, t) = W_\alpha f^{(eq)}(\vec{x}, \vec{e}_\alpha, t) = w_\alpha \rho \Big[1 + \frac{3(\vec{e}_\alpha \cdot \vec{u})}{c^2} + \frac{9(\vec{e}_\alpha \cdot \vec{u})^2}{2c^4} - \frac{3\vec{u}^2}{2c^2}\Big]$$

where the discrete directions

$$\vec{e}_\alpha = \begin{cases} (0,0), & \alpha = 0 \\ (cos\theta_\alpha, sin\theta_\alpha)c, & \alpha = 1,2,3,4 \\ \sqrt{2}(cos\theta_\alpha, sin\theta_\alpha)c, & \alpha = 5,6,7,8 \end{cases}$$

can also be denoted using their angles

$$\theta_\alpha = \begin{cases} \frac{(\alpha-1)\pi}{2}, & \alpha = 1,2,3,4 \\ \frac{(\alpha-5)\pi}{2} + \frac{\pi}{4}, & \alpha = 5,6,7,8 \end{cases}$$

One could think about the above mentioned weights as factors compensating for the different lengths of the velocity vectors by choosing different masses for the particles moving along the different directions.

[124] $\sqrt{3RT}$ is the mean thermal velocity in the kinetic theory of gases (section C.2.2).

E.1.2. Chapman-Enskog expansion

Apart from different length scales molecular dynamics problems involve multiple time scales as well: [103], [104]

- Particles interact in collisions with a duration τ_{col} of order $\tau_{col} \sim s/v$.
- The time in between collisions is characterised by the mean flight time $\tau_\mu \sim \lambda/v$.
- On a macroscale, given by the minimum hydrodynamic time scale (convection, diffusion) $t_{hyd} \sim min[t_{con} = L/U, t_{dif} = L^2/\nu]$ where $Re = t_{dif}/t_{con}$, this leads to hydrodynamic flow.

All time steps up to a single collision ($0 < t \leq \tau_{int}$) are assumed instantly by the Boltzmann equation. Processes happening on a time scale $\tau_{int} < t \leq \tau_\mu$ describe the relaxation to a local Maxwellian equilibrium distribution with a space and time dependent flow speed and the regime $\tau_\mu < t \leq \tau_h$ describes the macroscopic transport across the system on the hydrodynamic time scale that normally is of interest. In order to recover this macroscopic behaviour on a significantly larger time scale an asymptotic analysis has to be performed. The following derivation of the macroscopic equations from the Boltzmann equation outside equilibrium, using a multiple-scale *perturbation* technique, is referred to as Chapman-Enskog expansion. [15]
One could see the hydrodynamic scale as a mean field representation emerging from the *perturbation of the underlying kinetic equations* with different time scales using perturbation theory, a method generally applied in celestial mechanics. The expansion parameter ϵ is generally seen as the *Knudsen number Kn*, defining the departure from continuum-based mechanics.[125]
The sequential time step can be expanded in a *Taylor series* assuming $\Delta t \approx \epsilon$ to

$$f_\alpha(\vec{x} + \Delta t\, \vec{e}_\alpha, \vec{e}_\alpha, t + \Delta t) = \sum_{m=0}^{\infty} \frac{\epsilon^m}{m!} D_t^m f_\alpha(\vec{x}, \vec{e}_\alpha, t)$$

where $D_t = (\partial_t + \vec{e}_\alpha \cdot \nabla)$ is the material derivative and the populations can be rewritten using a perturbation series of the equilibrium distribution (appendix J) to

$$f_\alpha = \sum_{n=0}^{\infty} \epsilon^n f_\alpha^{(n)}.$$

All distribution functions are assumed to be functions of multiple time scales $f(t_0, t_1, t_2)$ with $t_1 = \epsilon t_0$ and $t_2 = \epsilon^2 t_0$ and therefore the derivative with respect to t can be rewritten applying the chain rule to[126]

$$\partial_t = \sum_{n=0}^{\infty} \epsilon^n \partial_{t_n}.$$

This ansatz can be inserted into the Lattice-Boltzmann equation

$$f_\alpha(\vec{x} + \Delta t\, \vec{e}_\alpha, \vec{e}_\alpha, t + \Delta t) - f_\alpha(\vec{x}, \vec{e}_\alpha, t) = \frac{1}{\tau} \left[f_\alpha^{(eq)}(\vec{x}, \vec{e}_\alpha, t) - f_\alpha(\vec{x}, \vec{e}_\alpha, t) \right]$$

[125]This can be reasoned due to the Knudsen number in the dimensionless Boltzmann equation (appendix C.3.5).
[126]One might evaluate the corresponding order of magnitude of other terms due to a corresponding analysis commonly found in fluid dynamics.

leading to the following equation

$$\sum_{m=0}^{\infty}\Big(\frac{\epsilon^m}{m!}D_t^m\sum_{n=0}^{\infty}\epsilon^n f_\alpha^{(n)}\Big)+\Big(\frac{1}{\tau}-1\Big)\sum_{n=0}^{\infty}\epsilon^n f_\alpha^{(n)}-\frac{1}{\tau}f_\alpha^{(eq)}=0$$

which can be rewritten to

$$\Big(1+\epsilon D_t^1+\frac{\epsilon^2}{2}D_t^2+\cdots\Big)(f_\alpha^{(0)}+\epsilon f_\alpha^{(1)}+\epsilon^2 f_\alpha^{(2)}+\cdots)-$$
$$-(f_\alpha^{(0)}+\epsilon f_\alpha^{(1)}+\epsilon^2 f_\alpha^{(2)}+\cdots)+\frac{1}{\tau}(f_\alpha^{(0)}+\epsilon f_\alpha^{(1)}+\epsilon^2 f_\alpha^{(2)}+\cdots)-\frac{1}{\tau}f_\alpha^{(eq)}=0$$

and further inserting the formula of the material derivative to

$$\Big[\epsilon\Big(\partial_{t_0}+\epsilon\partial_{t_1}+\cdots+\vec{e}_\alpha\cdot\nabla\Big)+\frac{\epsilon^2}{2}\Big(\partial_{t_0}+\cdots+\vec{e}_\alpha\cdot\nabla\Big)^2+\cdots\Big](f_\alpha^{(0)}+$$
$$+\epsilon f_\alpha^{(1)}+\epsilon^2 f_\alpha^{(2)}+\cdots)+\frac{1}{\tau}(f_\alpha^{(0)}+\epsilon f_\alpha^{(1)}+\epsilon^2 f_\alpha^{(2)}+\cdots)-\frac{1}{\tau}f_\alpha^{(eq)}=0.$$

Now the terms of each order of ϵ have to vanish according to the *fundamental theorem of perturbation theory*[127] leading to the following set of equations.

$$\begin{aligned}
\epsilon^0:&\quad f_\alpha^{(0)}=f_\alpha^{(eq)}\\
\epsilon^1:&\quad D_{t_0}f_\alpha^{(0)}=(\partial_{t_0}+\vec{e}_\alpha\cdot\nabla)f_\alpha^{(0)}=-\tfrac{1}{\tau}f_\alpha^{(1)}\\
\epsilon^2:&\quad \partial_{t_1}f_\alpha^{(0)}+(\partial_{t_0}+\vec{e}_\alpha\cdot\nabla)f_\alpha^{(1)}+\tfrac{1}{2}(\partial_{t_0}+\vec{e}_\alpha\cdot\nabla)^2 f_\alpha^{(0)}+\tfrac{1}{\tau}f_\alpha^{(2)}=0
\end{aligned}$$

where the equation for ϵ^2 can be rewritten using the one for ϵ^1 to

$$\partial_{t_1}f_\alpha^{(0)}+\frac{2\tau-1}{2\tau}D_{t_0}f_\alpha^{(1)}=-\frac{1}{\tau}f_\alpha^{(2)}.$$

The constrains regarding the hydrodynamic moments must be fulfilled by the terms of order zero whereas the higher order terms ($n\geq 1$) must be equivalent to zero:

$$\sum_\alpha f_\alpha^{(0)}\begin{bmatrix}1\\ \vec{e}_\alpha\end{bmatrix}=\begin{bmatrix}\rho\\ \rho\vec{u}\end{bmatrix}\qquad\qquad \sum_\alpha f_\alpha^{(n)}\begin{bmatrix}1\\ \vec{e}_\alpha\end{bmatrix}=\vec{0}$$

For evaluating the following discrete moments it is useful to examine the properties of the tensor $E^{(n)}$ describing the projection of a discrete velocity onto the axes of a coordinate system multiplied by the weight of the corresponding direction (therefore the odd terms are equal to the zero vector $E^{(2n+1)}=\vec{0}$),[128] which are terms that can also be found in the moments above

$$E^{(n)}=\sum_{\alpha\neq 0}w_\alpha e_{\alpha i_1}e_{\alpha i_2}\cdots e_{\alpha i_n}$$

where $e_{\alpha i}$ is the projection of the direction $\vec{e}_\alpha$ and the unity vector of the i-axis.
In the $D2Q9$ model $|e_{\alpha i}|$ is either c or 0 for $\alpha=1..4$ or c^2 for $\alpha=5..8$ and due to the orthogonality of the projections on different axes only the diagonal elements of the following terms are different from zero

[127]This can also be argued by dividing the equation by $\epsilon,\epsilon^2,\ldots$ and letting ϵ tend to zero.
[128]This requirement actually stems from kinetic theory. [1]

$$\sum_{\alpha=1}^{4} e_{\alpha i} e_{\alpha j} = 2c^2 \delta_{ij} \qquad \sum_{\alpha=5}^{8} e_{\alpha i} e_{\alpha j} = 4c^2 \delta_{ij}$$

$$\sum_{\alpha=1}^{4} e_{\alpha i} e_{\alpha j} e_{\alpha k} e_{\alpha l} = 2c^4 \delta_{ijkl} \qquad \sum_{\alpha=5}^{8} e_{\alpha i} e_{\alpha j} e_{\alpha k} e_{\alpha l} = 4c^4 \Delta_{ijkl} - 8c^2 \delta_{ijkl}$$

with the non-diagonal elements

$$\Delta_{ijkl} = \delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}$$

due to pairwise orthogonal projections.
This leads to the family of tensors[129]

$$E^{(0)} = \sum_{\alpha} w_\alpha = 1$$

$$E^{(2)} = \sum_{\alpha \neq 0} w_\alpha e_{\alpha i} e_{\alpha j} = \left(\frac{2}{9} + \frac{4}{36}\right) c^2 \delta_{ij} = \frac{c^2}{3} \delta_{ij}$$

$$E^{(4)} = \sum_{\alpha \neq 0} w_\alpha e_{\alpha i} e_{\alpha j} e_{\alpha k} e_{\alpha l} = \left(\frac{2}{9} - \frac{8}{36}\right) c^2 \delta_{ij} + \frac{4}{36} c^4 \Delta_{ijkl} = \frac{c^4}{9} \Delta_{ijkl}$$

With these properties of the tensors $E^{(n)}$ we can calculate the moments of the discrete equilibrium distribution function to:

$$\sum_{\alpha} f_\alpha^{(0)} = \rho,$$

$$\sum_{\alpha} e_\alpha f_\alpha^{(0)} = \rho \vec{u},$$

$$\sum_{\alpha} e_{\alpha i} e_{\alpha j} f_\alpha^{(0)} = \frac{c^2 \rho \delta_{ij}}{3} + \rho u_i u_j = \sigma_{ij}^{(0)},$$

$$\sum_{\alpha} e_{\alpha i} e_{\alpha j} e_{\alpha k} f_\alpha^{(0)} = \frac{c^2 \rho}{3} (\delta_{ij} u_k + \delta_{ki} u_j + \delta_{jk} u_i).$$

The first two moments of the equation of order ϵ^1 lead to the Euler equations

$$\partial_{t_0} \rho + \nabla \cdot (\rho \vec{u}) = 0,$$

$$\partial_{t_0} (\rho \vec{u}) + \nabla \cdot \sigma_{ij}^{(0)} = 0.$$

The moments of the equation of order ϵ^2 lead to equations

$$\partial_{t_1} \rho = 0$$

$$\partial_{t_1} (\rho \vec{u}) + \frac{(2\tau - 1)}{2\tau} \nabla \cdot \sigma_{ij}^{(1)} = 0$$

[129]Lattice discretisations have to fulfil these symmetries in order to reconstruct the Navier-Stokes equations on the continuum level.

where the term $\sigma_{ij}^{(1)}$ is given by

$$\sigma_{ij}^{(1)} = \sum_\alpha e_{\alpha i} e_{\alpha j} f_\alpha^{(1)} = -\tau \sum_\alpha e_{\alpha i} e_{\alpha j} D_{t_0} f_\alpha^{(0)} = -\tau[\partial_{t_0} \sigma_{ij}^{(0)} + \partial_{x_k} \sum_\alpha e_{\alpha i} e_{\alpha j} e_{\alpha k} f_\alpha^{(0)} =$$
$$= -\tau[\frac{-c^2}{3} \delta_{ij} \partial_{x_k}(\rho u_k) + \partial_{t_0}(\rho u_i u_j) + \partial_{x_k} \sum_\alpha e_{\alpha i} e_{\alpha j} e_{\alpha k} f_\alpha^{(0)}].$$

With

$$\partial_{t_0}(\rho u_i u_j) = u_i \partial_{t_0}(\rho u_j) + u_j \partial_{t_0}(\rho u_i) - u_i u_j \partial_{t_0} \rho =$$
$$= -u_i \partial_{x_k}(\frac{c^2}{3} \rho \delta_{jk} + \rho u_j u_k) - u_j \partial_{x_k}(\frac{c^2}{3} \rho \delta_{ik} + \rho u_i u_k) + u_i u_j \partial_{x_k}(\rho u_k) =$$
$$= -u_i \frac{c^2}{3} \partial_{x_j} \rho - u_j \frac{c^2}{3} \partial_{x_i} \rho - u_i \partial_{x_k}(\rho u_j u_k) - u_j \partial_{x_k}(\rho u_i u_k) + u_i u_j \partial_{x_k}(\rho u_k) =$$
$$= -u_i \frac{c^2}{3} \partial_{x_j} \rho - u_j \frac{c^2}{3} \partial_{x_i} \rho - \partial_{x_k}(\rho u_i u_j u_k)$$

and

$$\partial_{x_k} \sum_\alpha e_{\alpha i} e_{\alpha j} e_{\alpha k} f_\alpha^{(0)} = \partial_{x_k}[\frac{c^2}{3} \rho(\delta_{ij} u_k + \delta_{ik} u_j + \delta_{jk} u_i)] =$$
$$= \frac{c^2}{3} \delta_{ij} \partial_{x_k}(\rho u_k) + \frac{c^2}{3} \partial_{x_i} u_j + \frac{c^2}{3} \rho \partial_{x_i} u_j + \frac{c^2}{3} u_i \partial_{x_j} \rho + \frac{c^2}{3} \rho \partial_{x_j} u_i$$

we get

$$\sum_\alpha e_{\alpha i} e_{\alpha j} f_\alpha^{(1)} = -\tau[\frac{c^2}{3} \rho(\partial_{x_i} u_j + \partial_{x_j} u_i) - \partial_{x_k}(\rho u_i u_j u_k)].$$

Now combining the corresponding momenta according to their order of magnitude using $\partial_t = \partial_{t_0} + \epsilon \partial_{t_1}$ and including the equation above we yield

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0$$

and

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial}{\partial x_i}\left(\frac{c^2 \rho}{3}\right) + \frac{\partial}{\partial x_j}\left[\left(\tau - \frac{1}{2}\right)\left(\frac{c^2}{3}\rho\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) - \frac{\partial}{\partial x_k}(\rho u_i u_j u_k)\right)\right]$$

This resembles the familiar continuity and momentum equations if the viscosity is chosen as

$$\nu = \frac{c^2}{3}\left(\tau - \frac{1}{2}\right) = c_s^2\left(\tau - \frac{1}{2}\right)$$

where due to

$$-\frac{\partial}{\partial x_i}\left(\frac{c^2 \rho}{3}\right) = -\frac{\partial p}{\partial x_i}$$

pressure is linked to density via the aforementioned equation of state

$$p = c_s^2 \rho$$

and

$$-\frac{\partial}{\partial x_k}(\rho u_i u_j u_k)$$

is an error term emerging from the discretisation that is though of order $\mathcal{O}(Ma^3)$ and hence can be neglected for small local Mach numbers, if $\vec{u}$ is comparably small to the local speed of sound c_s.
This means the LBM model fulfils the *isothermal* Navier-Stokes equations for incompressible flow on the macro-scale if we choose the model parameters wisely and *combine numerical and physical viscosity* in a smart way but has left us at the same time with a couple of restrictions: We can only simulate weakly compressible fluids in the limit of small Mach and Knudsen numbers.[130] Additionally the kinematic *viscosity* has become a *lattice constant* and can't be adjusted independently any more. This may be problematic when the viscosity depends on external factors.
Due to these restrictions, the set of equations is generally used to simulate incompressible flows by choosing small local Mach numbers: In this case we are actually not solving the incompressible conservation equations directly but rather a *compressible, low Mach number approximation with its own lattice speed of sound* that allows for very large time steps and therefore high numerical efficiency. With this particular algorithm we are stuck with *uniform meshes* and therefore a proper resolution of the boundary layer will lead to a massive over-resolution everywhere else and thus to longer computation times.[131]
On the other hand the regular grid (therefore often referred to as "mesh-less") eliminates the biggest drawback in traditional Navier-Stokes based CFD, the meshing process. Furthermore the numerical scheme is fully explicit and as put by Sauro Succi *"non-linearity is local, non-locality is linear"*, meaning the most demanding step, the collision step is completely local making it perfectly suitable for *parallelisation* using large clusters leading to computational times unrivalled by traditional CFD method.

E.2. Force terms

External forces do not pose particular requirements for the conserved momenta and therefore might be modelled through an additionally artificial forcing term on the right hand side

$$-\frac{F_i}{m}\frac{(e_{\alpha i}-v_i)}{c_s^2}f_\alpha^{(eq)}$$

that does not contribute to the conservation of mass due to

$$-\frac{F_i}{m\,c_s^2}\sum_\alpha (e_{\alpha i}-v_i)f_\alpha^{(eq)}=0$$

but leads to a corresponding term in the momentum equation

$$-\frac{F_\alpha}{m\,c_s^2}\sum_\alpha (e_{\alpha i}e_{\alpha j}-v_i e_{\alpha j})f_\alpha^{(eq)}=-\rho\frac{F_i}{m}\delta_{ij}.$$

[130]This means no true incompressibility: Density changes are linked to pressure. This is a special case of the incompressible flow assumption.
[131]Advanced strategies like adaptive mesh refinement do exist though (appendix F).

E.3. Energy equation

For low Mach number flow the energy equation decouples from the other conservation equations and assuming an ideal gas with $h = c_P T$ one yields

$$\frac{\partial T}{\partial t} + u_j \frac{\partial T}{\partial x_j} = \frac{1}{\rho c_P} \frac{\partial}{\partial x_j} (k \frac{\partial T}{\partial x_j}).$$

Similarly to the Chapman-Enskog analysis for incompressible flow one may also derive a general algorithm for this type of *advection-diffusion*[132] equations

$$\frac{\partial(\rho\psi)}{\partial t} + \frac{\partial(\rho u_j \psi)}{\partial x_j} = \frac{\partial}{\partial x_j} (k \frac{\partial \psi}{\partial x_j}) + \dot{q}(\psi).$$

leading in case of the BGK collision operator to

$$g_\alpha(\vec{x} + \Delta t\, \vec{e}_\alpha, t + \Delta t) = g_\alpha(\vec{x}, t) + \omega_g \left[g_\alpha^{(eq)}(\vec{x}, t) - g_\alpha(\vec{x}, t) \right] + w_\alpha \dot{q}$$

with an equilibrium distribution

$$g_\alpha^{(eq)}(\vec{x}, t) = w_\alpha \psi \Big[1 + \frac{(\vec{e}_\alpha \cdot \vec{u})}{c_s^2} + \frac{(\vec{e}_\alpha \cdot \vec{u})^2}{2 c_s^4} - \frac{\vec{u}^2}{2 c_s^2} \Big].$$

and a collision frequency

$$\omega_g = \frac{c_s^2\, \Delta t}{k + \frac{\Delta t\, c_s^2}{2}}.$$

If a feedback is desired this is often modelled as a temperature-density coupling such as the Boussinesq approximation where the temperature-dependent fluid density is modelled as a buoyancy force instead of changing the simulation density.[5] Fully consistent thermal models that treat the total energy as a higher moment require higher order lattices such as multi-speed lattices which involves significantly more complex algorithms and in particular boundary conditions.

[132]Often convection and advection are used as synonyms but I think in this context advection is more suiting.

Appendix F.

LBM beyond incompressible flow

F.1. Derivation of Lattice-Boltzmann algorithms

The *Lattice Boltzmann methods* (LBM) could more generally be seen as *molecular dynamics algorithms* that can be *arbitrarily designed* in such a way to *fulfil certain transport equations on a macroscopical level*. First the Boltzmann equation has to be discretised in time and space gaining a discrete Lattice Boltzmann equation acting on a finite set of velocity directions called the *lattice*. Then the behaviour on macroscopic level can be recovered using some sort of *asymptotic analysis*. The chosen discretisation is though strictly linked to the preserved differential equations: Some might not be suitable for modelling certain phenomena.
As the traditional *collision operator* is unhandy it is replaced by collision operators modelling the effects of the collisions, which is the *relaxation to a equilibrium distribution*, rather than modelling the collision itself. In order to recover arbitrary differential equations hence one might employ a combination of different *equilibrium functions*, *collision operators* and *forcing terms*. It is even common to couple several *different lattices* solving distinct differential equations and linking them with adequate forcing terms.
Generally a set of *lattice units* is introduced that is convenient for the corresponding simulation leading to a simulation space Δx and time step Δt equal to 1. The link to any particular simulation has to be made using the law of similarity and the relevant *dimensionless numbers*.
Finally also the behaviour of *boundary conditions* has to be modelled on the level of distribution functions. As for the equation itself there exist *several different implementations* that all have certain advantages and drawbacks regarding locality, computing time and order of accuracy.

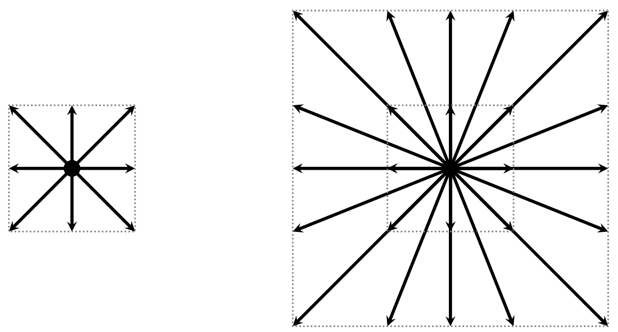


Figure F.1.: The popular $D2Q9$ lattice (left) and a rather exotic multi-speed lattice (right)

F.1.1. Finite-difference discretisations

Generally the Boltzmann equation is discretised with what could be seen as a finite-difference discretisation using lattices denoted by the common *DdQq* notation but there even exist exotics, called *multi-speed lattices* [48], that do not only interact with their direct neighbours but also with nodes one lattice apart (figure F.1).
In order to lift the restrictions of a uniform mesh a couple of mesh refinement techniques have been developed (figure F.2) [105] as well as techniques involving dynamically adapting mesh refinement [106], [107] introducing filtering as well as interpolation steps.

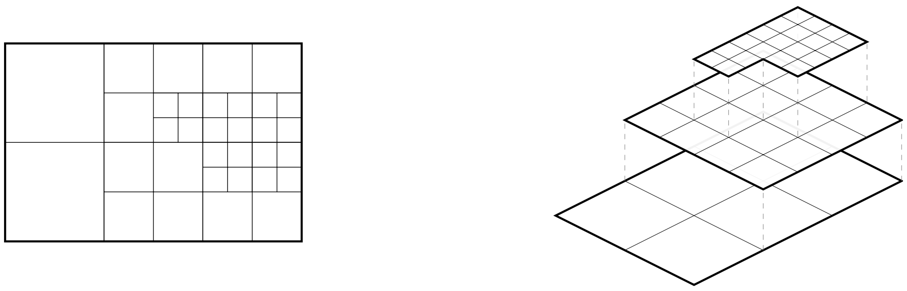


Figure F.2.: Schematic working principle of block structured meshes

F.1.2. Finite-volume discretisations

Even though still relatively rare it should not be neglected that even a few finite-volume discretisations exist that can be applied to unstructured grids. [108], [109]

F.2. Applications

Traditionally LBM is used in *computational fluid dynamics*, more particularly for *incompressible fluid flow*. Due to a low Mach number approximation of the Maxwell-Boltzmann distribution being employed this particular model introduces significant error for larger Mach numbers. But there have been made efforts for solving also thermal [48], compressible[133] or even supersonic [110], [111] as well as multi-phase and multi-component flow [112]. Additionally it has been tried to use similar algorithms in order to simulate other physical phenomena like radiation [113], [114] and the Schrödinger equation in Quantum mechanics. [115] This renders LBM a useful tool throughout computational physics in particular for stiff problems involving drastically different time scales and system where macroscopic equations do not exist. [63]

[133]Due to offering no real computational advantage and due to stability reasons these methods still lack significant advantage over traditional methods[1]

F.3. Generic advantages

As the operating level of any Lattice-Boltzmann algorithm is a level deeper than traditional differential equation solvers, one could argue a more fundamental level, namely on the level of distribution functions, it may be easier to implement certain *microscopic* phenomena. The main advantage though is the *computational performance*. Most popular schemes take a very favourable form: They are fully explicit, require only local information and therefore are ideal for *parallel implementations*. This is though at the same time also their biggest drawback. As the evolution equations are inherently time dependent they tend to be not particularly suited for steady state simulations.

Appendix G.

Computer architecture and programming

Writing a working LBM algorithm is quite straight forward but writing an efficient one is quite challenging. The fully explicit scheme in Lattice-Boltzmann, unlike in most other scientific computing methods, involves only light computational tasks like simple addition and multiplication but as the amount of grid cells is quite immense most of the time is spent copying values from the memory to the processor's cache. Therefore the main problem for a fast implementation is finding *fast containers* and a proper *memory layout* that makes use of the computer architecture. This appendix will address some basics of programming and computer architecture in general to further explain the reasoning behind the chosen implementation.

G.1. Basics of computer architecture

G.1.1. Virtual memory: Stack and heap

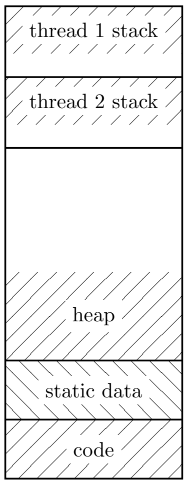


Figure G.1.: Memory organisation of a typical program

Generally the operating system manages the *physical memory* of a system and makes it accessible to applications by assigning them *virtual memory*. The application itself has no clue where the memory it is addressing is physically located, it only knows

the virtual memory address that is translated by the operating system to a physical memory address.[134]
At start-up each thread of a program gets assigned a certain *limited* small portion of main memory that can be managed very efficiently, called the *stack*. Additionally a larger portion of memory, the *heap*, can be addressed (figure G.1). Contrary to the stack it can *dynamically grow* within certain limits (see the next paragraph) but has to be managed manually: If memory that is not needed any more is not freed, this portion of memory can't be re-allocated during the runtime of the process (memory leak).

G.1.2. Processor architecture and word size

In computing a *word* is a fixed-size piece of data that is handled as a unit by the processor and widely affects the computer's operation. It is characterised by the number of bits it contains, which in modern computers is either 32 or 64.[135] As the format of the memory addresses between the two are different, 64-bit software is not compatible with a 32-bit architecture while forward compatibility is guaranteed.
The *virtual address size* is limited by the word: In 32-bit systems the address is a 32-bit value whereas in 64-bit systems it is a 64-bit value. This limits the amount of memory that can be allocated by a single 32-bit application to 4 GB. Depending on the operating system, part of it may be used for the kernel, further reducing to 3 GB (Linux) or 2 GB (Windows) (compare to figure G.1). Therefore in order to lift this restriction the application has to be compiled as 64-bit.[136]

G.1.3. Processor clock rate and Intel turbo boost

One of the main indicator for performance is the frequency a particular processor core is running at, the clock rate generally given in gigahertz GHz. Modern Intel processors offer a *dynamic over-clocking* feature named turbo boost that allows the processor frequency to be automatically raised when demanding tasks are run. When the workload rises the processor's clock is increased in increments[137] until it reaches its limit given by temperatures, power and current. Therefore the maximum clock frequency is a *function* of the *number of cores currently in use*.

G.1.4. Memory architecture

Within the last years processor performance has increased significantly while the speed of memory modules has only increased slowly: Without additional measures modern processors would idle most of their cycles waiting for new data to be loaded from the memory. The two characteristic numbers are the *latency*, which indicates the

[134]This also allows secondary storage to be accessed in a similar fashion in order to exceed the size of available physical memory (paging).
[135]Software specifically designed for such a hardware is referred to as 32- or 64-bit software.
[136]This may require an own compiler flag. For Windows even an own 64-bit compiler like TDM-GCC is required.
[137]Therefore for benchmarks a meaningful time frame has to be chosen: The processor will take some time to reach its turbo speed.

time between a data request and its completion, and the rate of information transfer, the *bandwidth*.[138]

G.1.4.1. Memory latency and multi-level cache

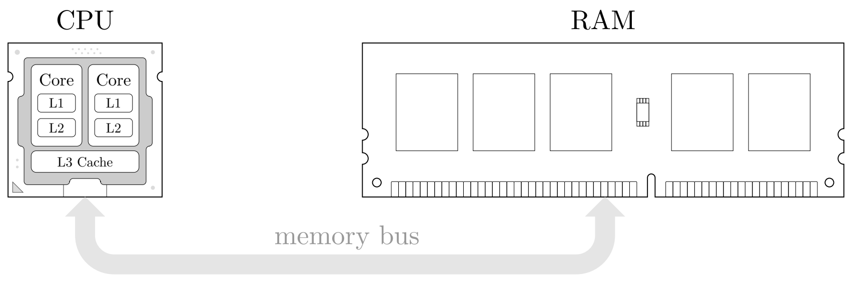


Figure G.2.: Schematic CPU cache hierarchy

Every time the CPU accesses the main memory (RAM)[139] a certain delay is introduced, which is referred to as *memory latency*. In order to hide this latency, small buffers, so called *caches*, are located on the CPU, which store previously used data. The smallest data unit that a cache can handle is the so called cache line[140]. When the access to a certain value in memory is requested a full cache line is loaded from main memory, meaning also nearby values are fetched into the cache in the hope that those will be used next.[141] Before requesting the next chunk of data from the memory the caches are searched, possibly boosting performance. If the required data can't be found (cache miss) the processor is forced to issue a main memory access with a much higher latency.
In modern computers there are different layers of cache that differ in size and speed (*multi-level cache*). While the first two layers, first- (L1) and second-level cache (L2), serve only a single core, the third-level cache (L3) is generally shared among all cores of a single processor (figure G.2). The cache higher up in the hierarchy tends to be faster while the cache on the lower end is generally by magnitudes larger.

G.1.4.2. Multi-channel hardware architecture

In order to boost the transfer rate between the memory and the processor[142] a multi-channel memory architecture employs multiple busses between them. This allows different memory controllers to access the modules in parallel theoretically multiplying the transfer rate by the number of channels but requires identical modules and a compatible architecture. While consumer grade platforms generally only support dual-channel memory[143] high-end processor micro-architectures use four memory channels (quad-channel).

[138]One would wish for vanishing latency and high bandwidth in order to yield optimal performance.
[139]This so called Von Neumann architecture has proven far from ideal but is historically grown.
[140]The corresponding cache line size varies with the architecture.
[141]Future computers might turn to smart AI pre-aching in order to boost performance.
[142]Precisely between the RAM and the corresponding memory controller
[143]This must be distinguished from dual data-rate memory (DDR) where data exchange happens twice per memory clock.

G.1.4.3. Data bandwidth

The data bandwidth of a system is equal to the product of

- the product of *base frequency* and the *number of data transfers per cycle*[144], which is generally indicated on the modules
- the *bus width*, which is 64 bits for all common modules
- the *number of channels*

If the memory is significantly slower than the processor, the latter has to wait and the processor time is wasted. In order to keep a processor constantly busy one should therefore try to keep the speed of the memory close to the processor clock speed and use the data stored in the caches.[145]

G.1.5. Multi-tasking and multi-threading

Traditional single-core processors can only run one instruction at a time. In order to allow multi-tasking, meaning multiple processes to be ran on a single core, each process has to divided into smaller subsets of instructions, so called threads, and then the different subsets have to be processed by the CPU consecutively according to a specified schedule (time-slicing). Systems with multiple processors or multiple cores on the other hand can handle different threads simultaneously distributing the load among the different cores (multi-threading).

G.1.5.1. Hardware multi-threading: Simultaneous multi-threading

A lot of high-end processors offer the ability of using hardware multi-threading: Each processor has *two virtual processor cores*[146] that can be addressed by the operating system. Another thread might use resources which are currently not used, doubling the simultaneous threads and theoretically also the performance. This is referred to as *simultaneous multi-threading* (SMT) while for Intel processors it is labelled hyper-threading and is reported to lead to a performance increase of up to 30%.[147]

G.2. Interpreted and compiled implementations

In order to be executed, a program has to be converted into machine language. This can be done in two main ways: *Compiled implementations* translate the whole source code into machine code beforehand while *interpreted implementations* execute the program directly line by line at run time, translating each command into subroutines that are already pre-compiled in machine code. While latter minimises platform dependency it tends to be significantly slower.

[144]The number of data transfers is two for the common case of DDR modules.

[145]LBM is generally bandwidth limited. Apart from re-using values stored in the caches one should favour multi-core set-ups with large caches and high-speed memory modules that allow for quad-channel usage. Else the cores might not get the data required leading to a weak parallel scaling.

[146]This requires an adequate processor architecture: Certain structures in a single core must duplicated.

[147]While for certain unoptimised processes it might even lead to reduced performance.

G.3. Matlab

Matlab is a *numerical computing environment* vastly used in engineering that employs a relatively easy proprietary programming language including features like graphic interfaces that allow for fast software prototyping. Partially based on LAPACK, a library for numerical linear algebra, it is optimised for matrix and vector operations. This requires a code to be fully vectorised in order to achieve optimal performance but leaves little options for the user to optimise their code by themselves. Even though Matlab can also be compiled using the Matlab Compiler SDK (requires the free library Matlab Runtime) code is generally interpreted and optimisation algorithms as well as external libraries calls introduce additional overhead. As a rule of thumb a well-written C or Fortran program can outperform Matlab code by a factor of 10-100x.

G.4. C++

C++ is a *general-purpose programming language*, developed from its ancestor C with performance in mind. It introduces several advanced features like object-orientation in the form of classes, inheritance, polymorphism and function overloading, while yet retaining the simplicity and efficiency of C code.
Being a low level programming language it leaves a lot of optimisation to the user. Some hints on writing efficient code are given in the paragraphs below: Some apply to C++ only while others may be applied to other programming languages like Fortran as well. First though we will have to take a look at what happens internally when compiling a C++ program in order to outline at which point we can influence the process.

G.4.1. Compilation in C++ and compiler settings

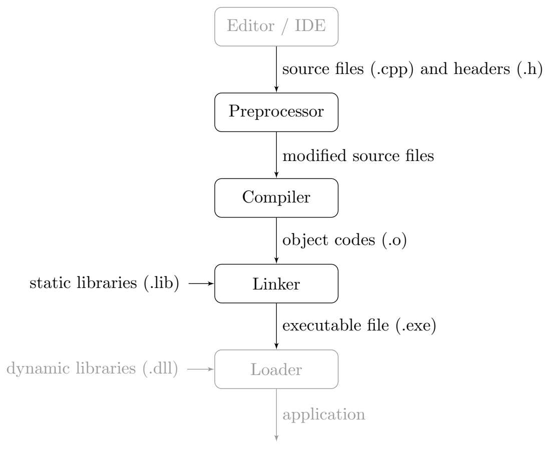


Figure G.3.: The compilation process in C++ on a Windows system

While simple code is generally written into a single source file, larger project are split up into several files. This aids understanding of isolated parts of the code and due to

a small change the whole project has not to be recompiled again but instead only the corresponding file.
If different source files interact with each other they have to be aware of the declaration of functions in other files they might access. Copying and pasting the corresponding definitions would be error prone and is therefore avoided by *separating the definition from the implementation*. This gives birth to so called header files that contain the function declaration and are included with a hint (preprocessor directives) to a special structure in the compilation process, the pre-processor. The pre-processor literally replaces the corresponding lines marked with an `#include` with the content of the header files or the corresponding libraries. In order to avoid multiple declarations each header file is generally assigned a *header guard*, a conditional that guarantees that a header is included only once - in the case its guard word hasn't been defined yet.
During the following *compilation process* (figure G.3) each source file is first *compiled* to relocatable machine instructions, a so called object file, and then those are *linked* together and combined with static libraries to a single executable. There are multiple compilers available involving additional optimisation algorithms where the *GNU compiler collection* (GCC) and the performance-optimised Intel C++ compiler (ICC) are probably the most common. While the first was initially written for GNU systems it has also been ported to other operating systems including Windows (MinGW and MinGW-w64). The compilation process itself can be influenced by supplying *compiler flags*.[148]

G.4.2. Object-oriented programming

C++ offers the possibility to add *blueprints for custom data types*, so called classes, that may include several variables or even functions. Other classes may inherit or even redefine parts of them (virtual functions).
Although this is very easy to understand and enables somebody to write very well structured code, it may introduce overhead as well as aggravate optimisation.[149] For this reason in this thesis a non-object orientated approach was taken as the first object-oriented prototype turned out to be significantly slower than expected.

G.4.3. Passing by value and by reference

Several programming languages including C++ offer the concept of *pointers*. A pointer is an object that simply refers to another value stored in the memory using its *memory address*. If a parameter is passed by value between two functions each function has its own independent local copy of the variable but if a parameter is passed by reference using a pointer both functions simply share the same address, meaning a variable can also be overwritten in a sub-function. For large arrays an intelligent use of this concept can significantly increase performance and memory consumption. In this thesis all arrays such as the populations and the indices of the boundary conditions are passed by reference using pointers.

[148] In this thesis the two flags `-O2` for code optimisation for speed as well as `-M64` for a compilation of 64-bit applications are used.

[149] And as LBM involves a huge number of steps per second a small overhead per step may lead to a significantly slower performance.

G.4.4. Choice of containers

In C++ there are two main containers that are used in scientific computing. An *array* is a fixed-size series of elements of the same data type placed in contiguous memory locations. If the array turns out too big to be fitted into the stack it has to be manually allocated in the heap using `malloc` (returns a pointer, pointing at the memory location of the first element in the heap, or a null pointer if the memory could not be allocated). This requires the user though to keep track of the array as well as its size manually (use `free` for de-allocation).[150] Else random locations of memory containing random data may be accessed.
Vectors (require the corresponding vector library to be included) do the same job as arrays but can be changed in size dynamically offering several functions that allow to reserve space and shrink it again as well as appending elements. Storage is handled automatically by the container itself introducing a small overhead.
Where performance is crucial and the size of the object foreseeable, an array should be used, for objects that change in size during their life-time a vector is able to deliver an inferior yet reasonable performance. In this thesis therefore the populations are allocated in the heap as arrays whereas for the boundary conditions indices that might change in size vectors are used.

G.4.5. Inline functions

If a function is called, the memory address of the corresponding instructions has to be saved, parameters have to be copied to the stack, the function code has to executed, return values have to be stored and the control has to be returned to the calling function. This introduces a small *overhead* that is irrelevant for computationally heavy functions but introduces massive overhead for small but frequently used functions where this switching time is relevant.[151]
In C++ it is possible to reduce this overhead by declaring functions as *inline functions*. If reasonable the compiler *substitutes the function call with the corresponding function* possibly increasing performance. This can be handy for repetitive tasks such as finding linear indices from two or three dimensional coordinates in our LBM simulation.[152]

G.4.6. Other optimisations

- The most basic but generally most effective way of optimising written code is choosing more *appropriate algorithms*: Different terms may be rephrased leading to mathematically easier expressions while repeating terms might be *calculated only once*, stored in a corresponding variable and should be *reused as soon as possible*.
- Where performance is crucial avoid fetching data from data structures and instead hard-code it.[153]
- Variables that do not change during a simulation should be marked as *constant* at declaration allowing for further compiler optimisation.

[150]One might use the C++ syntax `new` and `delete` instead but should not mix those two concepts.
[151]Therefore rather place loops inside functions than functions inside loops.
[152]If the same inline function should be used in external files it must be completely declared in the header file.
[153]Like index changes for boundary conditions.

- The *prefix operator* `++i` should be preferred over the postfix operator `i++` as for the latter a temporary copy has to be made.
- Every *loop* introduces some sort of overhead due to the required control instructions such as the pointer arithmetic and the "end of loop" tests after each iteration. This can be partially avoided by *unrolling* the corresponding loops: The loops are rewritten as a sequence of similar independent statements and the loop counter is adjusted accordingly. Like this, in case of a cache miss, out-of-order CPUs are able to continue calculations, potentially hiding the memory latency.
- For *file export* it might be a good idea to write to *binary files*, as writing in ASCII requires a format conversion, slowing down the export significantly.

G.4.7. Parallelisation and multi-threading

In the last years there has been a shift in consumer electronic systems away from single processor peak performance to the use of multiple cores: Instead of one strong processor core handling all of the tasks sequentially, a bunch of slower cores carrying out processes simultaneously has proved to be more economic.[154] This requires a task to be divided into smaller elementary operations that can be handled by single processors independently at the same time.
In order to compare the performance of sequential and parallel processes we define the *speed-up* $S(p)$ as the ratio between sequential runtime $T(1)$ to parallel runtime $T(p)$ where p is the number of processors:

$$S(p) = \frac{T(1)}{T(p)}.$$

Ideally one would wish for performance to scale linearly with the number of processors being used. In reality this isn't the case due to overhead and tasks that can only be executed sequentially and one additionally defines the *efficiency* $E(p)$ as the ratio between the reached and the ideal linear speed-up to

$$E(p) = \frac{S(p)}{p}.$$

The efficiency of a specific problem highly depends on the problem being solved: A high efficiency can be considered as a low *communication to computation* ratio: Calculations can be done locally and independently without having to rely on information from distant processing units. State of art algorithms reach an efficiency of over 80% (figure G.4).

Most common programming languages were released way before this trend of parallel computing could have been predicted: They work sequentially and have natively none or only very limited parallel capabilities. These features have to be unlocked by including additional application programming interfaces (API).
For systems where processors share the same memory (*shared memory*) usually *OpenMP*, an add-on to the compiler, is chosen. A single process is launched that can call *a number of subsets*, threads, which can be run by different processor cores (*multi-threading*). Parallelising a code using OpenMP is pretty straight forward, in relevant parts simply

[154]Supercomputers are based on the same strategy but on a more sophisticated level: Processors are combined to huge clusters (the currently most powerful supercomputer, IBM Sequoia, covers almost $300m^2$) splitting up the calculations between them.

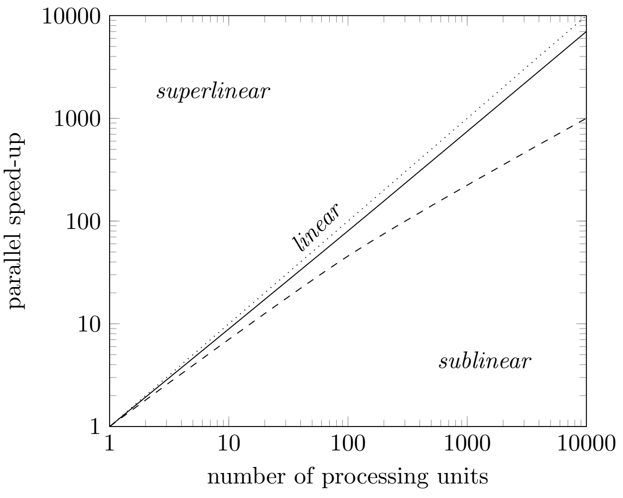


Figure G.4.: Ideal (dotted), high (solid) and low parallel efficiency (dashed)

additional comments to the pre-processor are added. The process should though be designed in such a way to ease the use of multiple threads such as maintaining data locality.
For *distributed systems* where a number of independent computer nodes interact with each other, a different approach has to be taken: Every parallel process works in isolation with its own memory and variables must be *kept in sync* with each other using *message-passing interfaces*, short *MPI*. This makes it a lot more complicated to program than in the case of a single shared memory.
Additionally *graphics cards* (GPUs) have proved to be very efficient as they are massive parallel structures by themselves already with thousands of small cores[155] that can handle instructions independently and connected by fast buses but come at a lower price tag and have a lower power consumption. Initially computational problems had to be reformulated in terms of graphic primitives but the increasing interest finally led to general purpose programming languages (GPGPU), including graphic card developer nVidia's proprietary Compute Unified Device Architecture (*CUDA*), a programming language based on C, first released in 2007, that allows GPUs to be abused for heavy calculations. This has caused an increasing interest of universities and companies in building their own 'low budget' clusters in order to speed up their calculations.[156]

[155]nVidia latest high end model Titan V has over 5000 cores.
[156]GPU programming might though be significantly harder than traditional shared (such as OpenMP) or distributed memory systems (OpenMPI): Possible optimisations depend significantly on the particular architecture and therefore requires specific knowledge about the individual GPU architecture. Parallelising large domains may though be challenging as graphics cards have a strictly limited memory size. In 2014 for this reason the concept of a single memory space, the so called Unified Memory was introduced. The overall performance of the algorithm is therefore mainly limited by the slow communication between different graphics cards in a setup.

G.4.8. Amdahl's law

For an process consisting of several functions the overall *speed-up* due to the speed-up s of a single function with the portion of computing time p can be calculated according to Amdahl's law to

$$speedup = \frac{runtime_{old}}{runtime_{new}} = \frac{1}{(1-p)+\frac{p}{s}}.$$

This is particularly useful in parallel computing for determining the possible speed-up if some functions only work sequentially.

G.4.9. OpenMP

```
#include <omp.h>

omp_set_nested(1);
unsigned int thread_max = omp_get_max_threads();
omp_set_num_threads(thread_max);

#pragma omp parallel sections
{
  #pragma omp section
  {
    ...   //task 1
  }
  #pragma omp section
  {
    ...   //task 2
  }
}
```

Figure G.5.: Schematic code of nested parallelism in OpenMP: The two tasks, task 1 and task 2, are handled at the same time. Due to the nested parallelism flag both tasks might involve parallelism themselves.

Natively C++ code is only sequential which is a huge drawback in times of massively parallel CPUs. One possible way of *parallelising* sequential code is by creating *multiple* small synchronised sequences that can be handled by a processor, called *threads* (multi-threading). In C++ this is generally done through the Open Multi-Processing API (OpenMP).
The implementation is pretty straight forward: The corresponding libraries have to be included (`#include <omp.h>`), the compiler has to be notified with the compiler flag `-fopenmp` and the linker has to be supplied with the corresponding library (`libgomp-1.dll` for 32bit or `libgomp_64-1.dll` for 64bit). Then the syntax, basically consisting of pre-compiler directives (`#pragma omp` ...) that handle thread creation as well as race conditions can be used. By default nested parallelism is not enabled but it might come in handy in some cases. Figure G.5 shows an OpenMP example with nested parallelism activated.

G.4.10. A note on platform dependencies and portability

C++ is per se platform independent but must be compiled individually for a particular target platform. When including external (platform dependent) libraries this might change though, restricting the code to a specific operating system. Particular care must be taken when implementing graphic interfaces as most current APIs (Win32, Qt, WxWidgets, GTK+, ...) are only compatible with a single or few different operating system.

Appendix H.

Benchmarks

The following chapter is a collection of benchmarks that were not considered suitable or important enough to be mentioned in the main part of the thesis.

H.1. Lid driven cavity: Pressure and vorticity

The following two curves were moved into the appendix as I have no benchmark data to compare them against. They show good visual agreeance and no artificial artefacts but I lack data to compare them against: Most visualisations do not mention the corresponding values of the iso-lines.

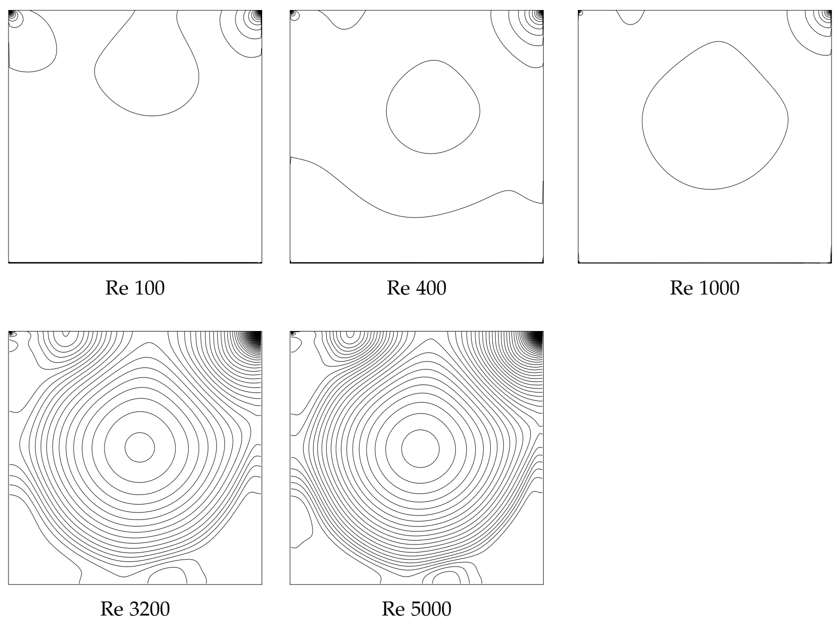


Figure H.1.: Pressure contours for a square LDC at different Reynolds numbers

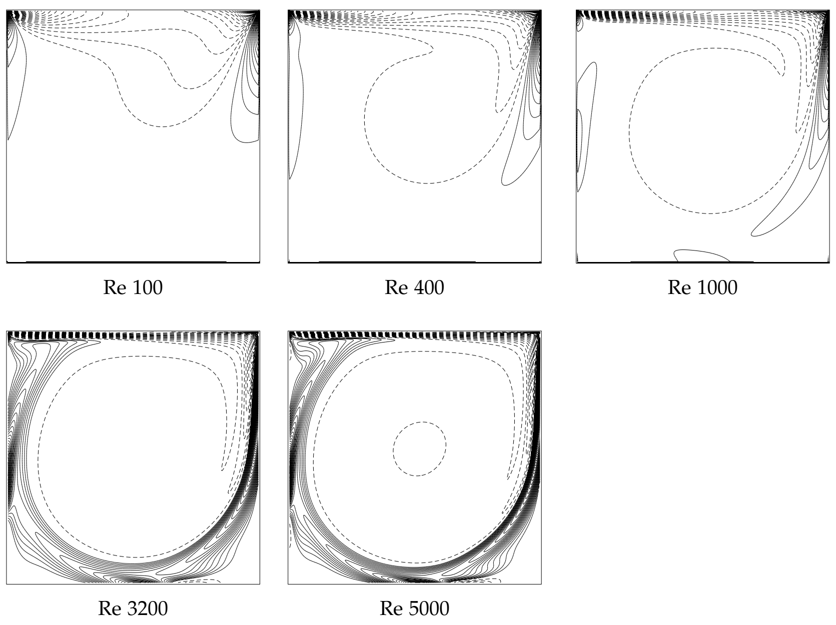


Figure H.2.: Vorticity contours for the case of a square LDC at different Reynolds numbers

H.2. Multi-component flow: Effects of wrong simulation parameters

The following example should illustrate what a huge *effect* the setting of *wrong parameters* might have. The characteristic length was wrongfully set to $L = 150$ rather than the correct length 146 and the inlet and outlet length were chosen to around 27% of the porous bed height rather than 1/3 as in the final simulation. Due to the small deviation the reader might think the results should be similar but this does not only lead to a shorter overall domain but also to a higher Reynolds number $Re = 150/146 \cdot 37.5 = 38.53$ and therefore also to a higher Peclét number. Thus the flow is *significantly more advection dominated* and additionally runs over a *shorter domain* length.

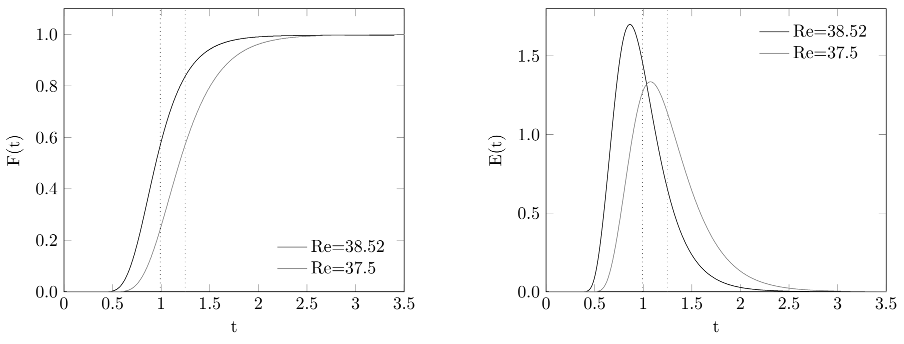


Figure H.3.: Residence time distribution (left) and its cumulative counterpart (right) for the faulty (black) and correct simulation (gray) as well as their mean residence times (dotted)

The hydrodynamic dwell time

$$\tau = \frac{V}{\dot{V}^{(in)}} = \frac{6881827}{6881827 + 3427635} \cdot \frac{600}{0.005} = 1.001s.$$

as well as the average residence time

$$\bar{t} = \int\limits_{t=0}^{\infty} t\, E(t)\, dt \approx \sum_i t_i\, E(t_i) = 0.9901$$

are significantly lower due to the smaller domain and stronger advection and the resident time distribution shows less dispersion (figure H.3).

Appendix I.

Mathematical appendix

This chapter tries to roughly outline the mathematical basics immediately needed in order to understand this thesis as well as to clarify the chosen notations. I tried to outline in particular quadrature rules as well as perturbation theory as I'd consider both hidden gems, rarely seen in engineering literature. The content is by no means complete and everybody who wants to know more about the particular topic should pick up the corresponding specialised literature.

I.1. Mathematical notations

I.1.1. Scalars

$\lvert x\rvert$	Absolute value	$\begin{cases} x, & \text{if } x \geq 0 \\ -x, & \text{if } x \leq 0 \end{cases}$
$\overline{x}$	Mean value	$\frac{x_1+\cdots+x_n}{n}$
$\langle x \rangle$	Weighted average	$\frac{w_1\,x_1+\cdots+w_n\,x_n}{w_1+\cdots+w_n}$
$f^{(n)}$	n-th term of perturbation analysis	
$n!$	Factorial	$\prod\limits_{i=1}^{n} i = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$

I.1.2. Vectors

$\vec{x}$	Column vector	$x_i = \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix}$
x_i	i-th component of the vector x	
$\vec{x} \cdot \vec{y}$	Dot product	$\langle x, y \rangle = x_j y_j = x_1 y_1 + \cdots + x_n y_n$
$\lvert\vec{x}\rvert$	Norm of the vector x	$\sqrt{x_1^2 + \cdots + x_n^2}$

$\vec{x}^2$	Square norm of the vector x	$x_1^2 + \cdots + x_n^2$
∇	Gradient	$\begin{bmatrix} \frac{\partial}{\partial x_1} \\ \cdots \\ \frac{\partial}{\partial x_n} \end{bmatrix}$
$\nabla_{\vec{v}}$	Gradient of $\vec{v}$	$\nabla \cdot \vec{v}$
∇^2	Laplace operator	$\sum\limits_{j=1}^{n} \frac{\partial^2}{\partial x_j^2}$

I.1.3. Matrices

$\underline{X}, X_{ij}$	Matrix	$X_{mn} = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \cdots & \cdots & \cdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$
X_{ij}	Matrix element	
$\underline{X}'$	Transposed matrix	$X_{nm} = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$

I.1.4. Differentials

D_t	Material derivative	$\frac{D}{Dt}$
∂_t	Local partial derivative	$\frac{\partial}{\partial t}$

I.1.5. Einstein notation

The *Einstein summation convention* is a mathematical notation according to which an index that appears twice in a single term is the short notation for a summation over this index. So the following two notations are equal:

$$a_j b_j \qquad \sum_j a_j b_j$$

I.1.6. Kronecker delta

The Kronecker delta is a function of two non-negative variables i and j. If the two variables are equal the function is 1, else it is 0.

$$\delta_{ij} = \begin{cases} 1, & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases}$$

I.2. Probability

I.2.1. Gaussian distribution function

For a large number of possible events a continuous distribution function, the so called Gaussian distribution function, can be used to predict the outcome.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-a)^2}{2\sigma^2}}$$

The values are arranged symmetrically around the mean value a with a standard deviation σ.

I.3. Integrals and derivatives

The following section outlines the basics of integral calculus, including important integrals used within the thesis and approximation methods, in particular Gaussian quadratures.

I.3.1. Material derivative

The rate of change of a continuum system $f(\vec{x}(t), t)$ that varies in space and time can be expressed, applying the chain rule

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial t} = \frac{\partial f}{\partial t} + u_j \frac{\partial f}{\partial x_j} = \frac{\partial f}{\partial t} + \vec{u} \cdot \nabla f.$$

I.3.2. Gauss's divergence theorem

The Gauss's divergence theorem describes the relation between the flux through a closed surface and the changes inside the volume enclosed by the surface.

$$\int_V \nabla \cdot \underline{T} dV = \oint_S \underline{T}' \cdot \vec{n} dA \qquad \int_V \frac{\partial T_{ij}}{\partial x_i} dV = \oint_S T_{ji} n_i dA$$

I.3.3. Important integrals

The following integral is necessary for evaluating the moments of distribution functions:

$$\int_{x=-\infty}^{\infty} \sqrt{\frac{C}{\pi}} e^{-Cx^2} dx = \left[\frac{1}{2}\,\mathrm{erf}\,(\sqrt{C}x)\right]_{x=-\infty}^{\infty} = 1$$

I.3.4. Orthogonal functions

Similarly to the dot product of two vectors, two functions $f(x)$ and $g(x)$ are considered orthogonal over an interval $a \leq x \leq b$ with a weighting function $w(x)$ if

$$\langle f, g\rangle = \int_a^b f(x)g(x)w(x)dx = 0$$

holds. A common example for orthogonal functions are $sin(x)$ and $cos(x)$ on the interval $-\pi \leq x \leq \pi$.

I.3.5. Gaussian quadrature

While most regular integrals may be evaluated numerically it might be necessary for computations of complex functions to approximate an integral numerically. A quadrature rule in numerical analysis is a method for *approximating a definite integral* of a function normally expressed as a weighted sum (weights w_i) of function values of particular points within the domain $f(x_i)$.
A polynomial of degree $n-1$ can be interpolated by n points and hence be integrated exactly with common quadrature rules. The Gaussian quadrature rule is a particular quadrature rule introduced by Carl Friedrich Gauss that yields an exact result with n points for polynomials of degree $2n-1$ or less by introducing the weights and the interpolating point locations (hence overall $2n$ measures compared to the n in basic quadrature rules) as unknowns:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i).$$

The weights w_i and abscisses x_i can be determined by different approaches (Gauss-Legrendre, Chebyshev-Gauss, Gauss-Hermite) but in either case they must be positive ($w_i > 0$) and possess following symmetries

$$x_i = -x_{n-i}, \qquad w_i = w_{n-i}, \qquad i = 1, 2, \cdots, n$$

Integrals on other finite intervals $[a, b]$ can be converted to integrals over $[-1, 1]$ by

$$\int_a^b f(t)dt = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{a+b+x(b-a)}{2}\right) dx$$

and hence also estimated by Gaussian quadrature.

I.3.5.1. Gauss-Hermite quadrature

The Gauss-Hermite quadrature is a special form of the Gaussian quadrature for approximating the type of integral

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

where the abscissas x_i and weights w_i for small n can be computed analytically to

n	x_1	x_2	x_3	w_1	w_1	w_3
2	$-\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$		$\frac{\sqrt{\pi}}{2}$	$\frac{\sqrt{\pi}}{2}$	
3	$-\sqrt{\frac{3}{2}}$	0	$\sqrt{\frac{3}{2}}$	$\frac{\sqrt{\pi}}{6}$	$\frac{2\sqrt{\pi}}{3}$	$\frac{\sqrt{\pi}}{6}$
...						

I.3.5.2. Two-dimensional Gauss product rules

The simplest form of bi-dimensional Gaussian quadrature rules, called product rules, deploy uni-dimensional quadrature rules along each natural coordinate.

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dx\,dy \approx \sum_{i=1}^{n}\sum_{j=1}^{m} w_i\,w_j\,f(x_i,y_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\,f(x_i,y_j)$$

Generally the same number of integration points per direction $n = m$ is used. In the case of a quadrature with $m = n = 3$ one yields the 9-point model depicted in figure I.1.

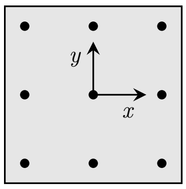


Figure I.1.: 9-point Gauss quadrature for rectangular elements

I.4. Series expansions

For complex functions it might be necessary to approximate a signal using a series expansion, a *sum of powers of one of its variables or other elementary functions*. Depending on the particular signal there are different series available, some of which will be described in the following section.

I.4.1. Taylor series

A Taylor series is a *representation of a function by an infinite sum of terms* calculated using the derivatives of a function evaluated in a certain point x_0. If this point is chosen to 0 the series is also referred to as *Maclaurin series*.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Generally just the n-th partial sum

$$T_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

is used to approximate a function locally.

I.4.2. Laurent series

A Laurent series expansion is a power series expansion that may be used to represent complex functions where a Taylor series expansion can't be applied such as for poles. In the case of e^x it is given by

$$e^x = \sum_{n=0}^{\infty} \frac{z^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \cdots \quad \text{for } -\infty < x < \infty$$

which can be rewritten for the case of $e^{-\frac{1}{x}}$ to

$$e^{-\frac{1}{x}} = \sum_{n=0}^{\infty} \frac{1}{n!\,(-z)^n} = 1 - \frac{1}{x} + \frac{1}{2\,x^2} - \frac{1}{6\,x^3} + \frac{1}{24\,x^4} + \cdots \quad \text{for } -\infty < x < \infty$$

I.4.3. Fourier Series

In case of a *periodic functions* instead of approximating it locally with the use of a Taylor series one might try to approximate it using periodic functions obtaining a periodic result.
A Fourier series makes use of the orthogonality of sine and cosine functions and approximates a function as the infinite sum of sines and cosines[157]

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n cos(nx) + \sum_{n=1}^{\infty} b_n sin(nx)$$

where the coefficients are given by

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx,$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)cos(nx)dx,$$

[157]Near discontinuities this might lead to overshooting, often termed Gibbs phenomena.

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) sin(nx) dx.$$

One might combine sine and cosine terms using Euler's formula $e^{ix} = cos(x) + i\,sin(x)$, where i is the imaginary number, to

$$f(x) = \sum_{-\infty}^{\infty} A_n e^{inx}$$

where

$$A_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx.$$

These integrals can be easily transformed to an arbitrary interval of length L with control variable x_1 using

$$x = \frac{\pi x_1}{L}.$$

Generally a Fourier series is not used to generate an approximate result of a given function but instead to *generate an interpolation function to a given discrete signal*. In this case sums replace the above integrals. Often only the amplitudes for certain frequencies are of interest as this reflects the frequency response of a system.

I.4.3.1. Fast Fourier transform

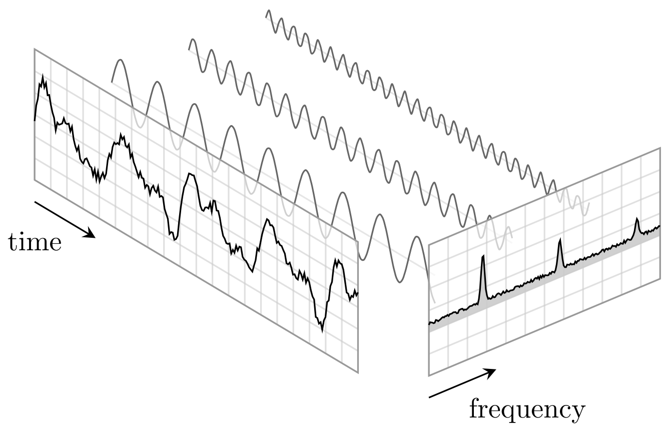


Figure I.2.: Schematic working principle of an FFT analysis: A signal over time is broken down into its frequency components

A fast Fourier transform (FFT) is an *algorithm* based on the Fourier series (section I.4.3) that *decomposes a signal over a period of time into its frequency components*, sinusoidal oscillations at different frequencies each with its own amplitude and phase (figure I.2). A few things should be kept in mind when working with FFT algorithms:

- A FFT is an *algorithm for periodic signals*, generally this means that the measuring interval is simply assumed to repeat itself. If the beginning and end of the signal do not match in phase this might introduce a phase change, if additionally the two amplitudes are not equal this basically introduces a Heaviside step function into the signal theoretically triggering indefinite[158] artificial higher frequencies. Using so called window functions the signal can be artificially forced to low values on each end of the signal similarly to a beat.

- Most numerical algorithms use a complex Fourier series that also includes the negative solutions which generally are of no physical importance. The resulting spectrum is therefore symmetric with regard to zero.
- One has to define a sample rate f_s which also dictates the highest frequency that can be detected by a FFT, the *Nyquist frequency* given by $0.5 f_s$, as well as the resolution of the analysis. For a finite time interval also nearby frequencies that are not present in the signal might be detected (spectral leakage).
- While it is generally known that a highest detectable frequency does exist, a lot of people are unaware that for a finite duration time window with N samples also a lowest frequency that can be detected exists, the so called *Rayleigh frequency* $\frac{1}{N}$, which corresponds to an oscillation with a wavelength spanning the entire sample interval.

[158]Theoretically indefinite but practically these artificial frequencies underlie the same limits as the signal itself and therefore the highest frequency is limited by the Nyquist frequency.

Appendix J.

Perturbation theory

Perturbation theory is a mathematical method for finding *approximate solutions to a variety of problems* including transcendental and differential equations. Starting with the exact solution of a simple unperturbed problem we expand it using a *power series in a small parameter ϵ* that quantifies the deviation from the exactly solvable problem x_0.

$$x = x_0 + \epsilon x_1 + \epsilon^2 x_2 + \cdots$$

Normally an approximate perturbation solution is obtained by *truncating the series* and usually keeping the first term only:

$$x \approx x_0 + \epsilon x_1$$

Now this can be applied to approximate a system where the exact solution to the unperturbed problem x_0 is known using the *fundamental theorem of perturbation theory*:

If

$$x_0 + \epsilon x_1 + \epsilon^2 x_2 + \cdots + \epsilon^n x_n + \mathcal{O}(\epsilon^{n+1}) = 0$$

for $\epsilon \to 0$ and $x_0, x_1, \cdots$ independent of ϵ, then

$$x_0 = x_1 = x_2 = \cdots = x_n = 0$$

For systems where this regular perturbation approach leads to *secular terms*, terms that can't be cancelled by choosing parameters accordingly and hence the solution grows without bound, there exist more sophisticated approaches like the *Poincaré-Linstedt method* (PLM) for periodic solutions and *multiple scale methods*.

J.1. Examples

As I think this topic is highly under-represented in the formation of any mechanical engineer this topic is supported by several examples starting with finding solutions to simple transcendental equations as well as examples for differential equations.

J.1.1. Example 1: Transcendental equations

Let's assume we want to find the solution to the transcendental equation

$$x - 2 = \epsilon\, cosh(x)$$

For $\epsilon \neq 0$ the equation above can't be solved in closed form. Yet for the unperturbed system ($\epsilon = 0$) the solution $x_0 = 2$ can be found.
Now we assume that for a small deviation ϵ we can write x as a power series $x = x_0 + \epsilon^1 x_1 + \epsilon^2 x_2 + \cdots$.
We insert this ansatz into the equation yielding

$$x_0 + \epsilon x_1 + \epsilon^2 x_2 + \cdots - 2 = \epsilon\, cosh(x_0 + \epsilon x_1 + \epsilon^2 x_2 + \cdots)$$

The terms of each order ϵ^n must vanish independently according to the fundamental theorem of perturbation theory. Taking into consideration the powerseries $cosh(x) = \sum \frac{x^{2n}}{(2n)!} = 1 + \frac{x^2}{2} + \cdots$ and therefore

$$\epsilon cosh(x_0 + \epsilon x_1) = \epsilon\Big[1 + \frac{(x_0 + \epsilon x_1)^2}{2} + \cdots\Big] = \epsilon\Big[1 + \frac{x_0^2 + 2\epsilon x_0 x_1 + \epsilon^2 x_1^2)^2}{2} + \cdots\Big]$$

the terms of order ϵ yield the equation

$$x_1 = cosh(x_0)$$

Hence we can approximate

$$x \approx x_0 + \epsilon x_1 = 2 + \epsilon\, cosh(2)$$

J.1.2. Example 2: Differential equations

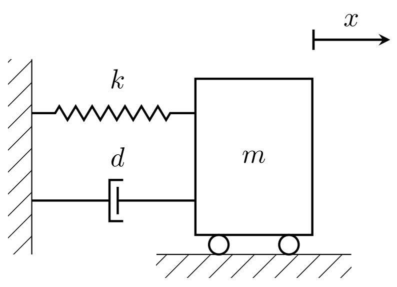


Figure J.1.: Mechanical model of an harmonic oscillator

Now we could use the same approach to solve differential equations e.g. the *Van der Pol equation* given by

$$\ddot{x} + \epsilon(x^2 - 1)\dot{x} + x = 0 \text{ with } \dot{x}(0) = 0$$

This is a non-linear second order differential equation describing the so called Van der Pol oscillator, a non-conservative oscillator with non-linear dampening. It can be seen as a small pertubation of the linear simple harmonic oscillator (figure J.1). As it is a non-linear equation an exact solution can't be found as matters stand but we may still

attempt to find an approximate solution using perturbation theory.
The unperturbed system ($\epsilon = 0$) has the solution $x_0(t) = A_0 sin(t) + B_0 cos(t)$ which simplifies due to the initial condition $\dot{x}(0) = 0$ to $x_0(t) = B_0 cos(t)$.
We assume that the full system will always remain pretty close to the linearized system but due to the non-linear terms periodicity is destroyed: The velocity of the oscillator and the shape of its orbit will be affected by the non-linearity.
We insert our ansatz up to order two into the differential equation yielding

$$\begin{aligned}(\ddot{x}_0 + \epsilon\ddot{x}_1 + \epsilon^2\ddot{x}_2) + \epsilon[(x_0 + \epsilon x_1 + \epsilon^2 x_2)^2 - 1](\dot{x}_0 + \epsilon\dot{x}_1 + \epsilon^2\dot{x}_2) + \\ + (x_0 + \epsilon x_1 + \epsilon^2 x_2) + \mathcal{O}(\epsilon^3) = 0\end{aligned}$$

This can be rewritten to

$$\ddot{x}_0 + x_0 + \epsilon[\ddot{x}_1 + x_1 + \dot{x}_0(x_0^2 - 1)] + \epsilon^2[\ddot{x}_2 + x_2 + \dot{x}_1(x_0^2 - 1) + 2x_0\dot{x}_0 x_1] + \mathcal{O}(\epsilon^3) = 0$$

As all x_i must be independent from ϵ the fundamental theorem of perturbation theory leads to

$$\begin{aligned}\epsilon^0: &\quad \ddot{x}_0 + x_0 = 0\\ \epsilon^1: &\quad \ddot{x}_1 + x_1 = \dot{x}_0(1 - x_0^2)\\ \epsilon^2: &\quad \ddot{x}_2 + x_2 = \dot{x}_1(1 - x_0^2) - 2x_0\dot{x}_0 x_1\end{aligned}$$

Now the ansatz is inserted in the equation of order ϵ^1 and rewritten using trigonometric identities to

$$\ddot{x}_1 + x_1 = \left(\frac{B_0^3}{4} - B_0\right) sin(t) + \frac{B_0^3}{4} sin(3t)$$

The solution of this differential equation can be found to

$$x_1(t) = -\left(\frac{B_0^3}{4} - B_0\right)\frac{t}{2} cos(t) - \frac{B_0^3}{32} sin(3t) + A_1 sin(t) + B_1 cos(t)$$

The first term in this equation is a *secular term* caused by the *resonating forcing term* of the differential equation[159]. It is unbounded for $t \to \infty$ and would clearly prevent a periodic solution. Therefore it has to be corrected manually to zero in order to force a periodic approximation by choosing B_0 accordingly. In this case that is the case for

$$B_0 = 2$$

Now we can insert this approximation in the equation of order ϵ^2 yielding

$$\ddot{x}_2 + x_2 = \frac{1}{4} cos(t) + 2B_1 sin(t) - \frac{3}{2} cos(3t) + 3B_1 sin(3t) + \frac{5}{4} cos(5t)$$

This differential equation can be solved to

$$x_2(t) = -\frac{3}{8} B_1 sin(3t) - B_1 t cos(t) + A_2 sin(t) + B_2 cos(t) + \frac{1}{8} t sin(t) + \frac{3}{16} cos(3t) - \frac{5}{96} cos(5t)$$

Now again this involves the secular term $\frac{1}{8} t sin(t)$ that can't be prevented as we have no free parameter left. We would need more free parameters for higher order approximations and therefore more sophisticated multiple scale methods have to be chosen.

[159]Terms with the same angular velocity as the solution of the homogeneous differential equation, in this case $sin(t)$ in the differential equation above

J.2. Two timing multiple scale method

The approximation of first order we derived so far breaks down if $t \geq \mathcal{O}(\epsilon^{-1})$. Due to accumulation of small effecs the amplitude of the oscillation is changed on a time scale ϵ^{-1} . We recognize there is actually two processes happening on two different time scales: We have the basic oscillation on a time scale of 1 and the slow drift in amplitude on a time scale ϵ^{-1}.
If a problem is characterized by *multiple physical processes each with their own scale* acting on a system at the same time it, makes sense to *introduce different time scales*. [116] Each variable is $\mathcal{O}(1)$ at its own relevant scale. In this case we introduce now a second time τ that accounts for the slow shift from the periodic solution. This so called 'slow time' τ can be defined in very different ways depending on the particular problem and thus the rate at which the non-linearity causes a drift in the orbit.
In our case we will define τ as

$$\tau = \epsilon\, t$$

With the ansatz from the regular perturbation theory now in two variables

$$x = x_0(t,\tau) + \epsilon x_1(t,\tau) + \epsilon^2 x_2(t,\tau) + \cdots$$

we can now write the Van der Pol equation

$$\frac{d^2x(t,\tau)}{dt^2} + \epsilon[x^2(t,\tau) - 1]\frac{dx(t,\tau)}{dt} + x(t,\tau) = 0$$

using

$$\frac{d\,x(t,\tau)}{d\,t} = \dot{x} + \epsilon\, x_\tau \quad \text{and} \quad \frac{d^2\,x(t,\tau)}{d\,t^2} = \ddot{x} + 2\epsilon\,\dot{x}_\tau + \epsilon^2 x_{\tau\tau}$$

to

$$\ddot{x} + 2\epsilon\,\dot{x}_\tau + \epsilon^2\,x_{\tau\tau} + \epsilon[x^2-1]\dot{x} + \epsilon^2[x^2-1]x_\tau + x = 0$$

Again we substitute the ansatz into the equation above and collect equal powers of ϵ:

$$\begin{aligned} \epsilon^0 : &\quad \ddot{x}_0 + x_0 = 0 \\ \epsilon^1 : &\quad \ddot{x}_1 + x_1 = \dot{x}_0(1 - x_0^2) - 2\dot{x}_{0\tau} \end{aligned}$$

This time we neglect the terms of $\mathcal{O}(\epsilon^2)$ as it gets quite complicated mathematically and the first approximation should be enough to illustrate the basic principle of this method.
Now we can obtain the general solution by integrating with respect to t, treating τ as an independent variable held constant and find the general solution

$$x_0 = C_0(\tau)cos[t + \theta_0(\tau)]$$

The functions C_0 and θ_0 are allowed to change at rates of the slow scale τ.
We assume now that the terms of order ϵ^0 fulfil the initial conditions and the other terms for each order are equivalent to zero respectively. It holds that

$$\frac{x}{d\,t} = \dot{x}_0(t,\tau) + \epsilon\,[x_{0\tau}(t,\tau) + \,\dot{x}_1(t,\tau)] + \epsilon^2[x_{1\tau}(t,\tau) + \dot{x}_2(t,\tau)] + \cdots$$

Due to the initial condition $\dot{x}_0(0) = 0$ we get $\theta_0(0) = 0$.
Inserting the solution for x_0 into the equation of order ϵ^1 yields

$$\ddot{x}_1 + x_1 = 2C_0\theta_{0\tau} cos[t + \theta_0] + (2C_{0\tau} + \frac{1}{4}C_0^3 - C_0) sin(t + \theta_0) + \frac{1}{4}C_0^3 sin[3(t + \theta_0)]$$

Now again we can integrate this equation and eliminate the response induced by the resonating forcing terms making use of the additional freedom due to the parameters $R(\tau)$ and $\theta_0(\tau)$. The resonating forcing terms $sin(t)$ and $cos(t)$ in the equation above have to be eliminated. This leads to the so called *solubility condition of Poincaré*:

$$2C_0\theta_{0\tau} = 0 \quad \text{and} \quad 2C_{0\tau} + \tfrac{1}{4}C_0^3 - C_0 = 0$$

With the initial conditions for the two variables C_0 and θ_0 that we obtained by looking at the terms of order ϵ^0 we can solve the differential equations above and get the two functions

$$\theta_0 = 0 \quad \text{and} \quad C_0 = \frac{2e^{\frac{\tau}{2}}}{\sqrt{1+e^{\tau}}}$$

Now we can solve the differential equation for x_1:

$$x_1 = -\frac{1}{32}C_0^3(\tau) sin(3t) + C_1(\tau) sin[t + \theta_1(\tau)]$$

Taking into consideration the conditions for order ϵ^1 we can now calculate the constants and looking at the terms of order ϵ^2 we can calculate the functions C_1 and θ_1 and continue like this for terms of higher order.
At higher orders the resonant forcing is unavoidable just like in the simple perturbation method as there can be insufficient freedom even with this ansatz. This problem can be overcome by introducing an even slower time scale $\mathrm{T} = \epsilon^2 t$.

J.2.1. Derivatives of multiple-scale perturbation series

For a function that can be written as a perturbation series and depends on multiple scales x_n

$$f(x) = f(x_0, x_1, x_2, \cdots) = f_0 + \epsilon f_1 + \epsilon^2 f_2 + \cdots$$

where x_n is given by

$$x_n = \epsilon^n x_0$$

the total derivative can be calculated according to the chain rule to

$$\frac{df}{dx} = \frac{df}{dx_0} + \epsilon \frac{df}{dx_1} + \epsilon^2 \frac{df}{dx_2} + \cdots = \sum_n \epsilon^n \frac{df}{dx_n}.$$

Bibliography

[1] D. Hänel, *Molekulare Gasdynamik: Einführung in die kinetische Theorie der Gase und Lattice-Boltzmann-Methoden*, en. Berlin Heidelberg: Springer-Verlag, 2004, ISBN: 978-3-540-44247-9 (cit. on pp. 3, 15, 16, 109, 113, 128, 134).

[2] M. Gad-el-Hak, "Questions in Fluid Mechanics: Stokes' Hypothesis for a Newtonian, Isotropic Fluid", *Journal of Fluids Engineering*, vol. 117, no. 1, pp. 3–5, Mar. 1995, ISSN: 0098-2202. DOI: 10.1115/1.2816816 (cit. on p. 4).

[3] G. Buresti, "A note on Stokes' hypothesis", *Acta Mechanica*, vol. 226, Oct. 2015. DOI: 10.1007/s00707-015-1380-9 (cit. on p. 4).

[4] S. R. Turns, *Thermodynamics: Concepts and Applications*, en. Cambridge University Press, Mar. 2006, Google-Books-ID: fy5hs04OeMQC, ISBN: 978-0-521-85042-1 (cit. on p. 8).

[5] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The Lattice Boltzmann Method - Principles and Practice*. Oct. 2016, ISBN: 978-3-319-44647-9. DOI: 10.1007/978-3-319-44649-3 (cit. on pp. 14, 18, 20, 32–34, 36, 38, 50, 51, 54, 70, 71, 73, 123, 132).

[6] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, English, 1 edition. Oxford : New York: Clarendon Press, Aug. 2001, ISBN: 978-0-19-850398-9 (cit. on pp. 15, 18, 41).

[7] T. G. Elizarova, *Quasi-Gas Dynamic Equations*, en, ser. Computational Fluid and Solid Mechanics. Berlin Heidelberg: Springer-Verlag, 2009, ISBN: 978-3-642-00291-5 (cit. on pp. 16, 21).

[8] G. Karniadakis, A. Beskok, and A. NR, *MicroFlows and Nanoflows - Fundamentals and Simulation*. Jan. 2005 (cit. on p. 16).

[9] S. A. Orszag and V. Yakhot, "Reynolds number scaling of cellular automaton hydrodynamics", eng, *Physical Review Letters*, vol. 56, no. 16, pp. 1691–1693, Apr. 1986, ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.56.1691 (cit. on p. 17).

[10] J. Hardy, O. de Pazzis, and Y. Pomeau, "Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions", *Phys. Rev. A*, vol. 13, May 1976. DOI: 10.1103/PhysRevA.13.1949 (cit. on p. 17).

[11] U. Frisch, B. Hasslacher, and Y. Pomeau, "Lattice-gas automata for the Navier-Stokes equation", eng, *Physical Review Letters*, vol. 56, no. 14, pp. 1505–1508, Apr. 1986, ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.56.1505 (cit. on p. 17).

[12] U. Frisch, D. Dhumieres, B. Hasslacher, P. Lallemand, Y. Pomeau, and J. P. Rivet, "Lattice Gas Hydrodynamics in Two and Three Dimensions", *Complex Systems*, vol. 1, Jan. 1987 (cit. on p. 17).

[13] A. A. Mohamad, *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*, English, 2011 edition. London ; New York: Springer, Apr. 2011, ISBN: 978-0-85729-454-8 (cit. on pp. 20, 27, 41).

[14] A. Wagner, "A Practical Introduction to the Lattice Boltzmann Method", (cit. on p. 21).

[15] S. Chapman, T. G. Cowling, and C. Cercignani, *The Mathematical Theory of Non-uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases*, English, 3 edition. Cambridge ; New York: Cambridge University Press, Jan. 1991, ISBN: 978-0-521-40844-8 (cit. on pp. 22, 127).

[16] C. Villani, *Entropy and H theorem: The mathematical legacy of Ludwig Boltzmann* (cit. on p. 22).

[17] E. G. D. Cohen, "Boltzmann and Statistical Mechanics", *arXiv:cond-mat/9608054*, Aug. 1996, arXiv: cond-mat/9608054 (cit. on p. 22).

[18] D. Tong, "University of Cambridge Graduate Course", en, p. 106, (cit. on p. 22).

[19] Q. Zou, S. Hou, S. Chen, and G. D. Doolen, "A improved incompressible lattice Boltzmann model for time-independent flows", en, *Journal of Statistical Physics*, vol. 81, no. 1-2, pp. 35–48, Oct. 1995, ISSN: 0022-4715, 1572-9613. DOI: 10.1007/BF02179966 (cit. on p. 22).

[20] P. Asinari, "Multi Scale Analysis of Heat and Mass Transfer in Mini/Micro Structures", PhD thesis, Politecnico di Torino, Turin, Feb. 2005 (cit. on p. 23).

[21] P. L. Bhatnagar, E. P. Gross, and M. Krook, "A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems", *Physical Review*, vol. 94, pp. 511–525, May 1954, ISSN: 1536-6065. DOI: 10.1103/PhysRev.94.511 (cit. on pp. 23, 24).

[22] H. Chen, S. Kandasamy, S. Orszag, R. Shock, S. Succi, and V. Yakhot, "Extended Boltzmann Kinetic Equation for Turbulent Flows", en, *Science*, vol. 301, no. 5633, pp. 633–636, Aug. 2003, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1085048 (cit. on p. 23).

[23] Y. Kuwata and K. Suga, "Lattice Boltzmann direct numerical simulation of interface turbulence over porous and rough walls", *International Journal of Heat and Fluid Flow*, SI\:TSFP9 special issue, vol. 61, no. Part A, pp. 145–157, Oct. 2016, ISSN: 0142-727X. DOI: 10.1016/j.ijheatfluidflow.2016.03.006 (cit. on p. 23).

[24] N. Pellerin, S. Leclaire, and M. Reggio, "An implementation of the Spalart–Allmaras turbulence model in a multi-domain lattice Boltzmann method for solving turbulent airfoil flows", *Computers & Mathematics with Applications*, vol. 70, no. 12, pp. 3001–3018, Dec. 2015, ISSN: 0898-1221. DOI: 10.1016/j.camwa.2015.10.006 (cit. on p. 23).

[25] M. Schreiber, P. Neumann, S. Zimmer, and H.-J. Bungartz, "Free-Surface Lattice-Boltzmann Simulation on Many-Core Architectures", *Procedia Computer Science*, Proceedings of the International Conference on Computational Science, ICCS 2011, vol. 4, no. Supplement C, pp. 984–993, Jan. 2011, ISSN: 1877-0509. DOI: 10.1016/j.procs.2011.04.104 (cit. on p. 23).

[26] M. Ashrafizaadeh and H. Bakhshaei, "A comparison of non-Newtonian models for lattice Boltzmann blood flow simulations", *Computers & Mathematics with Applications*, Mesoscopic Methods in Engineering and Science, vol. 58, no. 5, pp. 1045–1054, Sep. 2009, ISSN: 0898-1221. DOI: 10.1016/j.camwa.2009.02.021 (cit. on p. 23).

[27] J. Boyd, J. Buick, and S. Green, "A Second-Order Accurate Lattice Boltzmann Non-Newtonian Flow Model", *J. Phys. A: Math. Gen*, vol. 3950, Nov. 2006. DOI: 10.1088/0305-4470/39/46/001 (cit. on p. 23).

[28] J. Zudrop, K. Masilamani, S. Roller, and P. Asinari, "A robust lattice Boltzmann method for parallel simulations of multicomponent flows in complex geometries", *Computers & Fluids*, vol. 153, no. Supplement C, pp. 20–33, Aug. 2017, ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2017.04.021 (cit. on p. 23).

[29] N. Takada, M. Misawa, and A. Tomiyama, "A phase-field method for interface-tracking simulation of two-phase flows", *Mathematics and Computers in Simulation (MATCOM)*, vol. 72, no. 2, pp. 220–226, 2006, ISSN: 0378-4754 (cit. on p. 23).

[30] E. Fattahi, C. Waluga, B. Wohlmuth, U. Rüde, M. Manhart, and R. Helmig, "Lattice Boltzmann methods in porous media simulations: From laminar to turbulent flow", *Computers & Fluids*, vol. 140, no. Supplement C, pp. 247–259, Nov. 2016, ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2016.10.007 (cit. on p. 23).

[31] A. Masselot and B. Chopard, "A lattice Boltzmann model for particle transport and deposition", en, *EPL (Europhysics Letters)*, vol. 42, no. 3, p. 259, May 1998, ISSN: 0295-5075. DOI: 10.1209/epl/i1998-00239-3 (cit. on p. 23).

[32] X. Descovich, G. Pontrelli, S. Succi, S. Melchionna, and M. Bammer, "Modeling Elastic Walls in Lattice Boltzmann Simulations of Arterial Blood Flow", *IFAC Proceedings Volumes*, vol. 45, pp. 936–941, 2012. DOI: https://doi.org/10.3182/20120215-3-AT-3016.00165 (cit. on p. 23).

[33] X. He and N. Li, "Lattice Boltzmann simulation of electrochemical systems", *Computer Physics Communications*, vol. 129, no. 1, pp. 158–166, Jul. 2000, ISSN: 0010-4655. DOI: 10.1016/S0010-4655(00)00103-X (cit. on p. 23).

[34] S. Melchionna and S. Succi, "Lattice Boltzmann–Poisson method for electrorheological nanoflows in ion channels", *Computer Physics Communications*, Proceedings of the Europhysics Conference on Computational Physics 2004, vol. 169, no. 1, pp. 203–206, Jul. 2005, ISSN: 0010-4655. DOI: 10.1016/j.cpc.2005.03.045 (cit. on p. 23).

[35] H. Xu and Z. Dang, "Lattice Boltzmann modeling of carbon deposition in porous anode of a solid oxide fuel cell with internal reforming", *Applied Energy*, vol. 178, pp. 294–307, Sep. 2016, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2016.06.007 (cit. on p. 23).

[36] M. Sbragaglia and S. Succi, "A note on the lattice Boltzmann method beyond the Chapman-Enskog limits", en, *EPL (Europhysics Letters)*, vol. 73, no. 3, p. 370, Dec. 2005, ISSN: 0295-5075. DOI: 10.1209/epl/i2005-10404-8 (cit. on p. 23).

[37] F. Toschi and S. Succi, "Lattice Boltzmann method at finite Knudsen numbers", en, *EPL (Europhysics Letters)*, vol. 69, no. 4, p. 549, Jan. 2005, ISSN: 0295-5075. DOI: 10.1209/epl/i2004-10393-0 (cit. on p. 23).

[38] H. Grad, "Principles of the Kinetic Theory of Gases", en, in *Thermodynamik der Gase / Thermodynamics of Gases*, ser. Handbuch der Physik / Encyclopedia of Physics, Springer, Berlin, Heidelberg, 1958, pp. 205–294, ISBN: 978-3-642-45894-1 978-3-642-45892-7. DOI: 10.1007/978-3-642-45892-7_3 (cit. on p. 23).

[39] S. Ansumali and I. Karlin, "Single relaxation time model for entropic lattice Boltzmann methods", *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 65, p. 056 312, Jun. 2002. DOI: `10.1103/PhysRevE.65.056312` (cit. on p. 24).

[40] S. Chikatamarla, S. Ansumali, and I. Karlin, "Entropic Lattice Boltzmann Models for Hydrodynamics in Three Dimensions", *Physical review letters*, vol. 97, p. 010 201, Aug. 2006. DOI: `10.1103/PhysRevLett.97.010201` (cit. on p. 24).

[41] S. Chikatamarla and I. Karlin, "Entropic lattice Boltzmann method for turbulent flow simulations: Boundary conditions", *Physica A: Statistical Mechanics and its Applications*, vol. 392, pp. 1925–1930, May 2013. DOI: `10.1016/j.physa.2012.12.034` (cit. on p. 24).

[42] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, and L.-S. Luo, "Multiple-Relaxation-Time Lattice Boltzmann Models in Three Dimensions", *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 360, no. 1792, pp. 437–451, 2002, ISSN: 1364-503X (cit. on p. 24).

[43] L. Li, R. Mei, and J. F. Klausner, "Lattice Boltzmann models for the convection-diffusion equation: D2q5 vs D2q9", *International Journal of Heat and Mass Transfer*, vol. 108, pp. 41–62, May 2017, ISSN: 0017-9310. DOI: `10.1016/j.ijheatmasstransfer.2016.11.092` (cit. on pp. 24, 51).

[44] I. Ginzburg, "Two-relaxation-time Lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions", *Communications in Computational Physics*, vol. 3, pp. 427–478, Jan. 2008 (cit. on p. 25).

[45] M. Geier, A. Greiner, and J. Korvink, "Cascaded digital lattice Boltzmann automata for high Reynolds number flow", *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 73, p. 066 705, Jul. 2006. DOI: `10.1103/PhysRevE.73.066705` (cit. on p. 25).

[46] M. Geier, M. Schönherr, A. Pasquali, and M. Krafczyk, "The cumulant lattice Boltzmann equation in three dimensions: Theory and validation", *Computers & Mathematics with Applications*, vol. 70, no. 4, pp. 507–547, Aug. 2015, ISSN: 0898-1221. DOI: `10.1016/j.camwa.2015.05.001` (cit. on p. 25).

[47] D. Holman, R. Brionnaud, and Z. Abiza, "Solution to industry benchmark problems with the Lattice-Boltzmann code XFlow", *ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering, e-Book Full Papers*, pp. 6809–6824, Jan. 2012 (cit. on p. 25).

[48] N. Frapolli, S. S. Chikatamarla, and I. V. Karlin, "Multispeed entropic lattice Boltzmann model for thermal flows", *Physical Review E*, vol. 90, no. 4, p. 043 306, Oct. 2014. DOI: `10.1103/PhysRevE.90.043306` (cit. on pp. 27, 134).

[49] P. C. Philippi, L. A. Hegele Jr., L. O. E. d. Santos, and R. Surmas, "Deriving thermal lattice-Boltzmann models from the continuous Boltzmann equation: Theoretical aspects", *arXiv:physics/0506064*, Jun. 2005, arXiv: physics/0506064 (cit. on p. 27).

[50] J. Latt, "Hydrodynamic limit of lattice Boltzmann equations", eng, PhD thesis, University of Geneva, 2007 (cit. on pp. 27, 33).

[51] J. Latt, *Choice of units in lattice Boltzmann simulations*, Apr. 2008 (cit. on p. 27).

[52] H. Safari, M. Krafczyk, and M. Geier, "A Lattice Boltzmann model for thermal compressible flows at low Mach numbers beyond the Boussinesq approximation", *Computers & Fluids*, May 2018, ISSN: 0045-7930. DOI: `10.1016/j.compfluid.2018.04.016` (cit. on p. 34).

[53] S. Khirevich, I. Ginzburg, and U. Tallarek, "Coarse- and fine-grid numerical behavior of MRT/TRT lattice-Boltzmann schemes in regular and random sphere packings", *Journal of Computational Physics*, vol. 281, no. Supplement C, pp. 708–742, Jan. 2015, ISSN: 0021-9991. DOI: `10.1016/j.jcp.2014.10.038` (cit. on p. 34).

[54] I. Ginzburg, D. Dhumieres, and A. Kuzmin, "Optimal Stability of Advection-Diffusion Lattice Boltzmann Models with Two Relaxation Times for Positive/Negative Equilibrium", *Journal of Statistical Physics*, vol. 139, pp. 1090–1143, Jun. 2010. DOI: `10.1007/s10955-010-9969-9` (cit. on pp. 34, 51).

[55] R. Mei, L.-S. Luo, P. Lallemand, and D. d'Humières, "Consistent initial conditions for lattice Boltzmann simulations", *Computers & Fluids*, Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science, vol. 35, no. 8, pp. 855–862, Sep. 2006, ISSN: 0045-7930. DOI: `10.1016/j.compfluid.2005.08.008` (cit. on p. 35).

[56] S. Chen, D. Martínez, and R. Mei, "On boundary conditions in lattice Boltzmann methods", *Physics of Fluids*, vol. 8, no. 9, pp. 2527–2536, Sep. 1996, ISSN: 1070-6631. DOI: `10.1063/1.869035` (cit. on p. 35).

[57] T. Inamuro, M. Yoshino, and F. Ogino, "A non-slip boundary condition for lattice Boltzmann simulations", *Physics of Fluids*, vol. 7, no. 12, pp. 2928–2930, Dec. 1995, ISSN: 1070-6631. DOI: `10.1063/1.868766` (cit. on p. 39).

[58] Q. Zou and X. He, "On pressure and velocity flow boundary conditions and bounceback for the lattice Boltzmann BGK model", *Physics of Fluids*, vol. 9, no. 6, pp. 1591–1598, Jun. 1997, arXiv: comp-gas/9611001, ISSN: 1070-6631, 1089-7666. DOI: `10.1063/1.869307` (cit. on p. 39).

[59] M. Hecht and J. Harting, "Implementation of on-site velocity boundary conditions for D3q19 lattice Boltzmann", *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 01, P01018, Jan. 2010, arXiv: 0811.4593, ISSN: 1742-5468. DOI: `10.1088/1742-5468/2010/01/P01018` (cit. on pp. 39, 41).

[60] S. Hou, J. Sterling, S. Chen, and G. D. Doolen, "A Lattice Boltzmann Subgrid Model for High Reynolds Number Flows", *arXiv:comp-gas/9401004*, Jan. 1994, arXiv: comp-gas/9401004 (cit. on p. 44).

[61] H. Yu, S. S. Girimaji, and L.-S. Luo, "DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method", *Journal of Computational Physics*, vol. 209, no. 2, pp. 599–616, Nov. 2005, ISSN: 0021-9991. DOI: `10.1016/j.jcp.2005.03.022` (cit. on p. 44).

[62] M. Krafczyk, J. Tolke, and L.-S. Luo, "Large eddy simulation with a multiple-relaxation-time LBE model", *INTERNATIONAL JOURNAL OF MODERN PHYSICS B*, vol. 17, pp. 33–39, Jan. 2003. DOI: `10.1142/S0217979203017059` (cit. on p. 44).

[63] L.-S. Luo, "The Future of Lattice-Gas and Lattice Boltzmann Methods", en, in *Computational Aerosciences in the 21st Century*, ser. ICASE LaRC Interdisciplinary Series in Science and Engineering, Springer, Dordrecht, 2000, pp. 165–187, ISBN: 978-94-010-3807-2 978-94-010-0948-5. DOI: `10.1007/978-94-010-0948-5_9` (cit. on pp. 44, 134).

[64] V. Giovangigli, “Multicomponent Flow”, en, *Scholarpedia*, vol. 9, no. 4, p. 11 930, Apr. 2014, ISSN: 1941-6016. DOI: 10.4249/scholarpedia.11930 (cit. on p. 45).

[65] E. L. Cussler, *Diffusion: Mass Transfer in Fluid Systems*, en. Cambridge University Press, Jan. 2009, Google-Books-ID: dq6LdJyN8ScC, ISBN: 978-0-521-87121-1 (cit. on pp. 47, 55, 57).

[66] I. L. Mostinsky, “DIFFUSION COEFFICIENT”, in, Begellhouse. DOI: 10.1615/AtoZ.d.diffusion_coefficient (cit. on p. 47).

[67] B. Chopard, J. L. Falcone, and J. Latt, “The lattice Boltzmann advection-diffusion model revisited”, *European Physical Journal Special Topics*, vol. 171, pp. 245–249, Apr. 2009, ISSN: 1951-6355. DOI: 10.1140/epjst/e2009-01035-5 (cit. on pp. 49, 50).

[68] S. A. Hosseini, N. Darabiha, and D. Thévenin, “Mass-conserving advection-diffusion Lattice Boltzmann model for multi-species reacting flows”, *Physica A: Statistical Mechanics and its Applications*, vol. 499C, pp. 40–57, Jan. 2018. DOI: 10.1016/j.physa.2018.01.034 (cit. on pp. 49, 50).

[69] H.-B. Huang, X.-Y. Lu, and M. C. Sukop, “Numerical study of lattice Boltzmann methods for a convection–diffusion equation coupled with Navier–Stokes equations”, en, *Journal of Physics A: Mathematical and Theoretical*, vol. 44, no. 5, p. 055 001, 2011, ISSN: 1751-8121. DOI: 10.1088/1751-8113/44/5/055001 (cit. on pp. 51, 54).

[70] T. Zhang, B. Shi, Z. Guo, Z. Chai, and J. Lu, “General bounce-back scheme for concentration boundary condition in the lattice-Boltzmann method”, *Physical Review E*, vol. 85, no. 1, p. 016 701, Jan. 2012. DOI: 10.1103/PhysRevE.85.016701 (cit. on p. 52).

[71] L. Zhang, S. Yang, Z. Zeng, and J. W. Chew, “Consistent second-order boundary implementations for convection-diffusion lattice Boltzmann method”, *Physical Review E*, vol. 97, no. 2, p. 023 302, Feb. 2018. DOI: 10.1103/PhysRevE.97.023302 (cit. on p. 52).

[72] T. Gebäck and A. Geynts, “A Lattice Boltzmann Method for the Advection-Diffusion Equation with Neumann Boundary Conditions”, en, *Communications in Computational Physics*, vol. 15, no. 2, pp. 487–505, 2014, ISSN: 1815-2406. DOI: 10.4208/cicp.161112.230713a (cit. on p. 54).

[73] D. Wolf-Gladrow, “A lattice Boltzmann equation for diffusion”, en, *Journal of Statistical Physics*, vol. 79, no. 5, pp. 1023–1032, Jun. 1995, ISSN: 1572-9613. DOI: 10.1007/BF02181215 (cit. on p. 54).

[74] T. Xiao-Wu, S. Zu-Feng, and C. Guan-Chu, “Simulation of the relationship between porosity and tortuosity in porous media with cubic particles”, en, *Chinese Physics B*, vol. 21, no. 10, p. 100 201, 2012, ISSN: 1674-1056. DOI: 10.1088/1674-1056/21/10/100201 (cit. on p. 55).

[75] U. Ghia, K. N. Ghia, and C. T. Shin, “High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method”, *Journal of Computational Physics*, vol. 48, no. 3, pp. 387–411, Dec. 1982, ISSN: 0021-9991. DOI: 10.1016/0021-9991(82)90058-4 (cit. on p. 62).

[76] L.-S. Lin, Y.-C. Chen, and C.-A. Lin, "Multi relaxation time lattice Boltzmann simulations of deep lid driven cavity flows at different aspect ratios", *Computers & Fluids*, 22nd International Conference on Parallel Computational Fluid Dynamics (ParCFD 2010), vol. 45, no. 1, pp. 233–240, Jun. 2011, ISSN: 0045-7930. DOI: `10.1016/j.compfluid.2010.12.012` (cit. on p. 62).

[77] H. Ding, C. Shu, K. Yeo, and D. Xu, "Numerical computation of three-dimensional incompressible viscous flows in the primitive variable form by local multi-quadric differential quadrature method", *Computer Methods in Applied Mechanics and Engineering*, vol. 195, pp. 516–533, Jan. 2006. DOI: `10.1016/j.cma.2005.02.006` (cit. on p. 62).

[78] Z. Žunič, M. Hriberšek, L. Škerget, and J. Ravnik, "3d lid driven cavity flow by mixed boundary and finite element method", Jan. 2006 (cit. on p. 62).

[79] J. H. Lienhard, *Synopsis of lift, drag, and vortex frequency data for rigid circular cylinders*, ser. Bulletin ;300. Pullman: Technical Extension Service, Washington State University, 1966 (cit. on pp. 65, 66).

[80] E. Achenbach and E. Heinecke, "ON VORTEX SHEDDING FROM SMOOTH AND ROUGH CYLINDERS IN THE RANGE OF REYNOLDS NUMBERS 6 multiplied by 10**3 TO 5 multiplied by 10**6.", *Journal of Fluid Mechanics*, vol. 109, pp. 239–251, Aug. 1981 (cit. on pp. 65, 66).

[81] D. Hamane, O. Guerri, and S. Larbi, "A Comparative Study of Flow around a Circular Cylinder using Lattice Boltzmann Method", Jun. 2015. DOI: `10.2514/6.2015-3429` (cit. on p. 66).

[82] A. Roshko, *On the Development of Turbulent Wakes from Vortex Streets*, Report or Paper, 1954 (cit. on p. 68).

[83] A. Grucelski and J. Pozorski, "Lattice Boltzmann simulations of flow past a circular cylinder and in simple porous media", *Computers & Fluids*, vol. 71, pp. 406–416, Jan. 2013, ISSN: 0045-7930. DOI: `10.1016/j.compfluid.2012.11.006` (cit. on pp. 68, 69).

[84] F. Homann, "Einfluß großer Zähigkeit bei Strömung um Zylinder", de, *Forschung auf dem Gebiet des Ingenieurwesens A*, vol. 7, no. 1, pp. 1–10, Jan. 1936, ISSN: 0015-7899, 1434-0860. DOI: `10.1007/BF02578758` (cit. on p. 69).

[85] A. Thom, "The flow past circular cylinders at low speeds", en, *Proc. R. Soc. Lond. A*, vol. 141, no. 845, pp. 651–669, Sep. 1933, ISSN: 0950-1207, 2053-9150. DOI: `10.1098/rspa.1933.0146` (cit. on p. 69).

[86] L. Qu, C. Norberg, L. Davidson, S.-H. Peng, and F. Wang, "Quantitative numerical analysis of flow past a circular cylinder at Reynolds number between 50 and 200", *Journal of Fluids and Structures*, vol. 39, pp. 347–370, May 2013. DOI: `10.1016/j.jfluidstructs.2013.02.007` (cit. on p. 69).

[87] Wikichip, *Intel Core i9-7920x - WikiChip*, en (cit. on p. 79).

[88] J. Bear and Y. Bachmat, "Introduction to modeling of transport phenomena in porous media", English, Jan. 1990 (cit. on pp. 84, 90).

[89] J. M. Zalc, S. C. Reyes, and E. Iglesia, "The effects of diffusion mechanism and void structure on transport rates and tortuosity factors in complex porous structures", *Chemical Engineering Science*, vol. 59, no. 14, pp. 2947–2960, Jul. 2004, ISSN: 0009-2509. DOI: `10.1016/j.ces.2004.04.028` (cit. on p. 90).

[90] A. Koponen, M. Kataja, and J. Timonen, "Tortuous flow in porous media", *Physical Review E*, vol. 54, no. 1, pp. 406–410, Jul. 1996. DOI: 10.1103/PhysRevE.54.406 (cit. on p. 90).

[91] J. Comiti and M. Renaud, "A new model for determining mean structure parameters of fixed beds from pressure drop measurements: Application to beds packed with parallelepipedal particles", *Chemical Engineering Science*, vol. 44, no. 7, pp. 1539–1545, Jan. 1989, ISSN: 0009-2509. DOI: 10.1016/0009-2509(89)80031-4 (cit. on p. 90).

[92] Y. Bo-Ming and L. Jian-Hua, "A Geometry Model for Tortuosity of Flow Path in Porous Media", en, *Chinese Physics Letters*, vol. 21, no. 8, p. 1569, 2004, ISSN: 0256-307X. DOI: 10.1088/0256-307X/21/8/044 (cit. on p. 90).

[93] C. M. P. Caltech, Ed., *Equilibrium Versus Nonequilibrium* (cit. on p. 111).

[94] N. Borghini, *Equilibrium distributions* (cit. on p. 111).

[95] P. Asinari, "Multi-species Lattice Boltzmann Models and Practical Examples", (cit. on p. 115).

[96] W. He, W. Lv, and J. H. Dickerson, "Gas Diffusion Mechanisms and Models", en, in *Gas Transport in Solid Oxide Fuel Cells*, ser. SpringerBriefs in Energy, W. He, W. Lv, and J. Dickerson, Eds., Cham: Springer International Publishing, 2014, pp. 9–17, ISBN: 978-3-319-09737-4. DOI: 10.1007/978-3-319-09737-4_2 (cit. on p. 119).

[97] X. He and L.-S. Luo, "Lattice Boltzmann Model for the Incompressible Navier–Stokes Equation", en, *Journal of Statistical Physics*, vol. 88, no. 3-4, pp. 927–944, Aug. 1997, ISSN: 0022-4715, 1572-9613. DOI: 10.1023/B:JOSS.0000015179.12689.e4 (cit. on p. 122).

[98] X. He and L.-S. Luo, "A priori derivation of the lattice Boltzmann equation", *PHYSICAL REVIEW E*, vol. 55, R6333–R6336, Jun. 1997. DOI: 10.1103/PhysRevE.55.R6333 (cit. on p. 122).

[99] J. Li, "Appendix: Chapman-Enskog Expansion in the Lattice Boltzmann Method", *arXiv:1512.02599 [physics]*, Dec. 2015, arXiv: 1512.02599 (cit. on p. 122).

[100] L.-S. Luo, "Theory of the lattice Boltzmann method: Lattice Boltzmann models for nonideal gases", *Physical Review E*, vol. 62, no. 4, pp. 4982–4996, Oct. 2000. DOI: 10.1103/PhysRevE.62.4982 (cit. on p. 122).

[101] L. S. García-Colín, R. M. Velasco, and F. J. Uribe, "Beyond the Navier–Stokes equations: Burnett hydrodynamics", *Physics Reports*, vol. 465, no. 4, pp. 149–189, Aug. 2008, ISSN: 0370-1573. DOI: 10.1016/j.physrep.2008.04.010 (cit. on p. 124).

[102] n. Lallemand and n. Luo, "Theory of the lattice boltzmann method: Dispersion, dissipation, isotropy, galilean invariance, and stability", eng, *Physical Review. E, Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 61, no. 6 Pt A, pp. 6546–6562, Jun. 2000, ISSN: 1063-651X (cit. on p. 124).

[103] N. N. Bogoljubov, *Problems of a Dynamical Theory in Statistical Physics: N.N. Bogoliubov [Nikolaj Nikolaevič Bogoljubov]. Transl. from the Russian by E.K. Gora*, en. Geophysics Research Directorate, AF Cambridge Research Laboratories, Air Force Research Division, United States Air Force, 1960, Google-Books-ID: I5U8HAAACAAJ (cit. on p. 127).

[104] R. L. Liboff, *Kinetic Theory: Classical, Quantum, and Relativistic Descriptions*, English, 3rd edition. New York: Springer, Sep. 2003, ISBN: 978-0-387-95551-3 (cit. on p. 127).

[105] D. Lagrava, O. Malaspinas, J. Latt, and B. Chopard, "Advances in multi-domain lattice Boltzmann grid refinement", *Journal of Computational Physics*, vol. 231, no. 14, pp. 4808–4822, May 2012, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2012.03.015 (cit. on p. 134).

[106] A. Fakhari and T. Lee, "Finite-difference lattice Boltzmann method with a block-structured adaptive-mesh-refinement technique", *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 89, p. 033 310, Mar. 2014. DOI: 10.1103/PhysRevE.89.033310 (cit. on p. 134).

[107] A. Fakhari and T. Lee, "Numerics of the lattice boltzmann method on nonuniform grids: Standard LBM and finite-difference LBM", *Computers & Fluids*, vol. 107, pp. 205–213, Jan. 2015, ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2014.11.013 (cit. on p. 134).

[108] H. Xi, G. Peng, and S.-H. Chou, "Finite-volume lattice Boltzmann method", *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, vol. 59, pp. 6202–5, Jun. 1999. DOI: 10.1103/PhysRevE.59.6202 (cit. on p. 134).

[109] D. V. Patil, "Chapman–Enskog analysis for finite-volume formulation of lattice Boltzmann equation", *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 12, pp. 2701–2712, Jun. 2013, ISSN: 0378-4371. DOI: 10.1016/j.physa.2013.02.016 (cit. on p. 134).

[110] Y. Feng, P. Sagaut, and W.-Q. Tao, "A compressible lattice Boltzmann finite volume model for high subsonic and transonic flows on regular lattices", *Computers & Fluids*, vol. 131, pp. 45–55, Jun. 2016, ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2016.03.009 (cit. on p. 134).

[111] G. Trapani, D. M. Holman, R. Brionnaud, and O. Sosa, "A Supersonic Lattice-Boltzmann Method: Validation and Applications", in *35th AIAA Applied Aerodynamics Conference*, ser. AIAA AVIATION Forum, American Institute of Aeronautics and Astronautics, Jun. 2017. DOI: 10.2514/6.2017-4460 (cit. on p. 134).

[112] L.-S. Luo, "Unified Theory of Lattice Boltzmann Models for Nonideal Gases", *Physical Review Letters*, vol. 81, no. 8, pp. 1618–1621, Aug. 1998. DOI: 10.1103/PhysRevLett.81.1618 (cit. on p. 134).

[113] P. Asinari and R. Borchiellini, "A Lattice Boltzmann Formulation for the Analysis of Radiative Heat Transfer Problems in a Participating Medium", *NUMERICAL HEAT TRANSFER PART B-FUNDAMENTALS*, vol. 57, pp. 1–21, Mar. 2010. DOI: 10.1080/10407791003613769 (cit. on p. 134).

[114] A. Gairola and H. Bindra, "Lattice Boltzmann method for solving non-equilibrium radiative transport problems", *Annals of Nuclear Energy*, vol. 99, pp. 151–156, Jan. 2017, ISSN: 0306-4549. DOI: 10.1016/j.anucene.2016.08.011 (cit. on p. 134).

[115] S. Succi and R. Benzi, "Lattice Boltzmann equation for quantum mechanics", *Physica D: Nonlinear Phenomena*, vol. 69, no. 3, pp. 327–332, Dec. 1993, ISSN: 0167-2789. DOI: 10.1016/0167-2789(93)90096-J (cit. on p. 134).

[116] E. J. Hinch, *Perturbation Methods*, English. Cambridge ; New York: Cambridge University Press, Oct. 1991, ISBN: 978-0-521-37897-0 (cit. on p. 161).